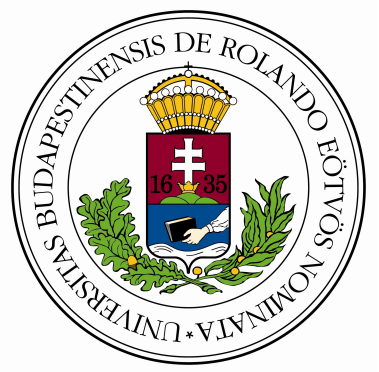
Eötvös Loránd Tudományegyetem
Természettudományi Kar

Kiss Bendegúz
Matematika BSc, alkalmazott matematikus szakirány

Kihagyásos túrák és fák

Szakdolgozat

Témavezető: Dr. Király Tamás
Operációkutatási Tanszék



Budapest, 2024

Tartalomjegyzék

Köszönetnyilvánítás

Szeretném megköszönni a témavezetőmnek, Király Tamásnak az egész féléves segítségét. Nagyon sok érdekeset tanulhattam a konzultációk közben amellett, hogy a szakdolgozat megírásához is jó útmutatást kaptam. Külön szeretném megköszönni családomnak és barátaimnak, hogy végig támogattak engem.

1. Bevezetés

A szakdolgozatom célja bemutatni néhány olyan problémát, amelyek az utazóügynök probléma továbbgondolásából születtek. Nyilván ezek is NP-nehéz problémák lesznek, így ezek megoldására is közelítő algoritmusokat vagy heurisztikákat érdemes keresni.

A szakdolgozat elején a Lagrange-relaxált témakörével fogunk foglalkozni. Ez azért indokolt, mert a k-MST probléma esetében egy nagyon szép alkalmazását fogjuk látni, amiből látszani fog a Lagrange-relaxált erőssége, ami nagyjából annyit jelent, hogy sok feltétel lazításának ellenére is elég jó megoldást fogunk kapni.

Ezt követően az első probléma, amivel foglalkozom a PTSP probléma (Penalty Travelling Salesman Problem), ahol a feladatunk abban különbözik az eredeti TSP feladattól, hogy nem muszáj minden csúcsot meglátogatni, de a kihagyott csúcsokért büntetést kell fizetnünk. A cél a túra költségének és a büntetéseknek a minimalizálása. Ennek megoldására fogunk látni egy primál-duál alapú 2-közelítő algoritmust, amely Goemans és Williamson nevéhez fűződik. A PTSP megoldásához definiáljuk majd a PCSTP problémát (Prize-Collecting Steiner-Tree Problem), ami hasonló a PTSP-hez csak ebben az esetben fát keresünk.

A második témakör a k-MST (k-Minimum Spaning Tree) és a k-TSP (k-Travelling Salesman Problem) probléma tárgyalásával kapcsolatos. Itt lényegében a feladatunk egy minimális költségű k csúcsból álló fát illetve túrát keresni. Ezekre egy 5-approximációs algoritmust fogunk adni, amely Garg nevéhez fűződik. A közelítés értékét lényegesen lecsökkentették 2.5-re, mégpedig úgy, hogy Garg algoritmusát szubrutinként alkalmazzák. Ez a módszer részletesen megtekinthető a [4] cikkben.

Végül a Quota-MST és a Quota-TSP problémákkal foglalkozzunk, ahol ugyanúgy mint eddig minimális költségű fát illetve túrát kell keresnünk. A nehezítés most annyi, hogy van egy súlyfüggvényünk, ami minden csúcshoz hozzárendel egy egész számot. Ezenkívül van egy előírt kvótánk is, ami egy egész szám. A feladat egy olyan fát találni, ami összegyűjti a megkövetelt kvótát, és ezen belül minimális költségű. Ezek megoldására Garg algoritmusát kicsit módosítjuk így jutva el az előző probléma esetén látott 5-közelítéshez. Emellett megvizsgáljuk, hogy a problémákra adható mohó algoritmusok mennyire működőképesek. Említésszerűen tárgyaljuk még a PCTSP (Prize-Collecting Travelling Salesman Problem) problémát is, ami lényegében a PTSP és a Quota-TSP problémák összevonása.

A problémákat elsősorban elméleti szempontból vizsgáljuk. A gyakorlati alkalmazásokra részletesen nem térünk ki. Megemlíthető, hogy a PCSTP és a PTSP probléma gyakorlati szempontból nem túl jelentős, viszont felhasználható egy gazdasági szempontból sokkal fontosabb problémára. Ez a NW maximalizálás (Net Worth Maximization), aminél ugyanaz az inputunk mint a PCSTP esetében csak itt a büntetéseket díjként kezeljük, és a célunk ezekből minél többet összegyűjteni, de közben figyelünk arra, hogy minél kevesebb legyen a megoldásunk költsége. Megfigyelhető, hogy a két probléma optimalizálás szempontjából ekvivalens, hiszen ha vesszük a megoldásaik összköltségét, akkor pontosan a csúcsok összsúlyát kell megkapnunk. Approximációs szempontból nézve, viszont egyáltalán nem ekvivalens a két probléma, hiszen már a NW maximalizálás közelítése is NP-nehéz, viszont a PCSTP-re létezik 2-approximációs algoritmus.

Nézzünk egy gyakorlati alkalmazást. Gondolhatunk itt egy internethálózat kiépítésére, ahol a potenciális ügyfelek a gráfunk csúcsai, az élek pedig a köztük esetlegesen kiépítendő internetkábelek. Az élek költségei megfeleltethetőek a kábelek hosszának, a csúcsokra írt díjak pedig a potenciális vevőknél várható nyereséget jelentik. Az internet-szolgáltató feladata ekkor egy olyan internethálózat (gráfelméletben egy fa) kiépítése, ami profitot termel, azaz minél kevesebb kábel használatával, minél több helyre eljutatja az internetet. Erről az alkalmazásról többet is olvashatunk a [9] cikkben. A k-MST vagy a Quota-MST alkalmazása hasonló az előbb említetthez. Ebben az esetben, viszont úgy akarunk kiépíteni egy ilyen hálózatot, hogy bizonyos számú ügyfélhez mindenképpen el kell jutnunk, mert anélkül nem lehetne beindítani a vállalkozást. Természetesen arra is törekszünk, hogy ezt minél olcsóbban tudjuk megoldani.

2. Fontosabb definiciók

Lássunk most néhány fontosabb definiciót mielőtt még elkezdenénk a problémáink tárgyalását. Ezek ismeretére mindenképpen szükségünk lesz később.

2.0.1. Definíció: (α-approximációs algoritmus)**.** *Az A polinomiális algoritmust α-approximációs algoritmusnak nevezzük, ha minden G inputra, amihez létezik megoldás, az A által szolgáltatott output költsége legfeljebb α-szorosa az optimum költségének, azaz: $c_A(G) \leq \alpha \cdot c_{OPT}(G)$. Itt α függhet az input méretétől is.*

2.0.2. Definíció:. *A $G = (V, E)$ teljes gráfon értelmezett $c : E \mapsto \mathbb{R}$ költségfüggvény teljesíti a háromszög-egyenlőtlenséget, ha $\forall u, v, w \in V$ esetén: $c(uw) \leq c(uv) + c(vw)$.*

2.0.3. Definíció:. *Egy $G = (V, E)$-beli sétát Euler-(kör)sétának nevezünk, ha a gráf minden élét tartalmazza és mindegyiket csak egyszer használja.*

2.0.4. Definíció:. *Legyenek $f, g : \mathbb{N} \mapsto \mathbb{R}^+$ függvények. Ha léteznek c, N konstansok, úgy hogy minden $n \geq N$ esetén $f(n) \leq c \cdot g(n)$, akkor azt mondjuk, hogy $f = O(g)$.*

2.0.5. Definíció: (TSP-feladat)**.** *Adott egy $G = (V, E)$ teljes gráf az élein egy $c : E \mapsto \mathbb{R}^+$ költségfüggvénnyel, amely teljesíti a háromszög-egyenlőtlenséget. A feladatunk találni egy minimális költségű $T = (V_T, E_T)$ Hamilton-kört.*

2.0.6. Megjegyzés. *A feladat definiálható összefüggő gráfra is, mert ekvivalens feladatot kapunk, ha megkonstruáljuk a gráf ún. metrikus lezártját, ami egy olyan G' teljes gráf, ami az eredeti gráfunk csúcsait tartalmazza és egy él költsége a végpontok között vezető legrövidebb út hossza G-ben.*

3. A Lagrange-relaxált

Mielőtt még a fent említett problémáinkkal foglalkoznánk, meg kell említenünk egy nagyon fontos fogalmat, amit az operációkutatásban sűrűn használnak, alkalmaznak. A témakör körbejárása után két alkalmazását is megmutatjuk. Az egyiket még ebben a fejezetben, ami csak egy picit kapcsolódik a témakörünkhöz, illetve majd a k-MST során is látunk egy ilyet.

3.1. A Lagrange-feladat

Először is vegyük a következő IP programot:

$$Qx \leq q \tag{3.1}$$

$$Ax \leq b$$

$$OPT_{IP} = \max cx$$

$$x \in \mathbb{Z}^n$$

Itt Q, A, q, b, c minden koordinátája egész. A következőkben azon dolgozunk, hogy az $Ax \leq b$ feltételt ne követeljük meg szigorúan, de így is releváns eredményt kapjunk. Ehhez definiáljuk a következő absztrakt megoldáshalmazt:

$$X = \{x \in \mathbb{Z}^n : Qx \leq q\}$$

Most tegyük fel, hogy adott c lineáris célfüggvény esetén X-en viszonylag könnyen számolunk optimumot(pl. minimális költségű feszítőfa). Ezt annak érdekében tesszük, hogy rámutassunk az eljárás gyakorlati hasznára. A másik feltételt X-hez hozzávéve a feladatunk nagyon bonyolulttá is válhat, ezért azon megpróbálunk valahogy lazítani. Nem csak annyit fogunk tenni, hogy egyszerűen elhagyjuk az $Ax \leq b$ feltételt, hanem az őt sértő $x \in X$ megoldásokat büntetni is fogjuk. Ehhez veszünk egy $\lambda \geq 0$ vektort, aminek dimenziója megegyezik b dimenziójával, és ennek rögzítésével definálhatjuk a következő ún. Lagrange-feladatot:

$$L(\lambda) := \max\{cx + \lambda(b - Ax) : x \in X\} \quad \lambda \geq 0 \tag{3.2}$$

Látható, hogy ha valamilyen i koordinátára $a_i x > b_i$, akkor λ_i-t minél nagyobbnak választva nem a maximalizálás irányába haladunk. Ha pedig $\lambda = 0$, akkor az azzal ekvivalens, hogy teljesen elhagyjuk a második feltételt.

Tetszőleges $\lambda \geq 0$ esetén megfigyelhető, hogy $OPT_{IP} \leq L(\lambda)$, hiszen tágabb megoldáshalmazzal dolgozunk.

A legjobb felső korlát megkeresésének érdekében definálhatjuk a Lagrange-feladat duálját:

$$OPT_{LD} = \min\{L(\lambda) : \lambda \geq 0\} \tag{3.3}$$

A (3.3) feladat megoldására ismertetjük majd a szubgradiens módszert a következő részben egy konkrét példa esetén. A k-MST során szintén látunk majd egy módszert az optimális λ megtalálására, ami egy bináris keresésen alapszik.

3.1.1. Megjegyzés. *Az itt elhangzottak hasonlóan működnek, ha $Ax = b$ alakú a relaxált feltételünk csak ebben az esetben tetszőleges λ-ra számolunk maximumot, minimalizálás esetén pedig minimumot.*

3.2. Egy alkalmazás: Held-Karp becslés

Ebben a részben megpróbálunk a TSP feladat megoldására minél jobb alsó becslést találni a Lagrange-relaxált alkalmazásával. Írjuk fel először is a TSP-hez tartozó IP-feladatot:

$$\sum_{e \in d(v)} x_e = 2 \quad \forall v \in V \tag{3.4}$$

$$\sum_{e \in \delta(U)} x_e \geq 1 \quad \emptyset \neq \forall U \subset V \tag{3.5}$$

$$x_e \in \{0, 1\}$$

$$OPT_{TSP}(c) = \min \sum_{e \in E} c(e) x_e$$

Itt x_e az élekhez tartozó bináris változó az alapján lesz 0 vagy 1, hogy bekerüle a túrába (ha igen x_e =1 különben $x_e = 0$). Egy ilyen élhalmaz akkor megengedett megoldás, ha minden csúcs foka 2 ez a (3.4) feltétel, ami ahhoz kell, hogy kört kapjunk, illetve minden valódi részhalmazba legalább egy él lépjen be (itt $\delta(U)$ az U-ba lépő élek halmaza) ez pedig a (3.5) feltétel, ami garantálja az összefüggőséget.

Legyen X azon 0-1 vektorok halmaza, amelyek (3.5)-t teljesítik. Most pedig relaxáljuk a (3.4) feltételt egy $\lambda = (\lambda_1 \dots \lambda_n)$ vektorral, amiből a következő feladatot kapjuk:

$$L(\lambda) = \min\{cx + \lambda\,(Ax - b) : x \in X\}$$

Itt A a gráfunk incidenciamátrixa és b a csupa 2 vektor. Most nézzük meg mit jelent $L(\lambda)$, ha kicsit átalakítjuk:

$$L(\lambda) = \min \left\{ \sum_{uv \in E} (c(uv) + \lambda_u + \lambda_v) x_{uv} : x \in X \right\} - 2 \sum_{v \in V} \lambda_v$$

Rögzített λ mellett a feladatunk ebben az esetben tehát egy minimális költségű összefüggő gráfot keresni egy új $c'(uv) = c(uv) + \lambda_u + \lambda_v$ költségfüggvény mellett. Mivel λ tetszőleges, lehetnek negatív élek, így úgy járunk el, hogy a negatív élekből alkotott komponenseket összehúzzuk egy csúccsá és a keletkező G' gráfban keresünk minimális feszítőfát, amire számos algoritmus létezik. A fent említettek alapján pedig a következő becsléshez jutunk:

$$TSP_{OPT}(c) \geq L(\lambda)$$

Erősebb alsó becsléshez jutunk az ún. 1-Fa becslés esetében. Ehhez vegyünk egy tetszőleges v_0 csúcsot. A (3.5) feltételt változtassuk meg a következőre:

$$\sum_{e \in \delta(U)} x_e \geq 2 \quad \emptyset \neq \forall U \subset V \setminus \{v_0\}$$

Látható, hogy a feladatunk ekvivalens marad az előzőhöz, hiszen egy TSP megoldás esetén legalább két élnek is be kell lépnie minden csúcshalmazba. Emellett kell legalább két él, ami kilép a $V \setminus \{v_0\}$ halmazból ezek csak v_0-ból jöhetnek és v_0 fokszáma miatt pontosan kettő ilyen lesz. Erre majd azért lesz szükségünk, hogy eggyel kevesebb feltételt tudjunk relaxálni. Ehhez viszont egy kicsit át kell fogalmaznunk a feladatot az ekvivalencia megőrzése mellett:

$$\sum_{e \in d(v)} x_e = 2 \quad \forall v \in V \setminus \{v_0\} \tag{3.6}$$

$$\sum_{e \in d(v_0)} x_e = 2 \tag{3.7}$$

$$\sum_{e \in E[U]} x_e \leq |U| - 1 \quad \emptyset \neq \forall U \subset V \setminus \{v_0\} \tag{3.8}$$

$$\sum_{e \in E[V \setminus \{v_0\}]} = |V| - 2 \tag{3.9}$$

$$\min \sum c(e) x_e$$

Itt $E[U]$ az U-ban kifeszített élhalmazt jelöli. Az első két feltétel lényegében a (3.4) szétválasztása illetve a (3.8) és a (3.9) együttesen kiadja az ún. feszítőfapoliédert $V \setminus \{v_0\}$-on, amelynek egész megoldásai a $V \setminus \{v_0\}$ csúcshalmaz feszítőfáihoz tartozó él-incidenciavektorok, a tört megoldásai meg ezen vektorok konvex burka. Részletesebben (3.8) gondoskodik arról, hogy egyik részgráfja se tartalmazzon kört $V \setminus \{v_0\}$-nak, és (3.9) pedig a feszítőfához szükséges élszámmal foglalkozik, ami jelen esetben $|V| - 2$. Ezek együttesen szintén a TSP feladat megoldásait szolgáltatják. Most pedig relaxáljuk a (3.6)-s feltételt és nézzük meg, hogy a minimalizálás része, hogy változik ekkor.

$$L'(\lambda) = \min \left\{ \sum_{uv \in E} c(uv)x_{uv} + \sum_{uv \in E[V \setminus \{v_0\}]} (\lambda_u + \lambda_v)x_{uv} + \sum_{v \in N(v_0)} \lambda_v x_{v_0 v} \right\} - 2 \sum_{v \in V} \lambda_v$$

Itt $N(v)$ a v csúcs szomszédjait jelöli. Kicsit átalakítva és felhasználva a már fent említett c' költségfüggvényt, azt kapjuk hogy:

$$L'(\lambda) = \min \left\{ \sum_{uv \in E[V \setminus \{v_0\}]} c'(uv)x_{uv} + \sum_{v \in N(v_0)} (c(v_0 v) + \lambda_v)x_{v_0 v} \right\} - 2 \sum_{v \in V} \lambda_v$$

Feltehető, hogy $\lambda_{v_0} = 0$, hiszen ez a megoldáshalmazon nem változhat illetve (3.7) megléte miatt a következő egyszerűbb alakra hozható $L'(\lambda)$:

$$\min \left\{ \sum_{uv \in E[V \setminus \{v_0\}]} c'(uv)x_{uv} \right\} + \min_{v_1, v_2 \in N(v_0)} \{c'(v_0 v_1) + c'(v_0 v_2)\} - 2 \sum_{v \in V} \lambda_v$$

A két minimalizálási részt együttesen 1-Fa becslésnek nevezzük, ami most a c' célfüggvény mellett értendő. A gráfunkon a következő történik ezen becslés során. Veszünk egy tetszőleges v_0 csúcsot majd keresünk egy minimális költségű feszítőfát a $V \setminus \{v_0\}$ csúcshalmazon és hozzávesszük a v_0-ból kilépő élek közül a két legkisebb költségűt. Ezeknek az összköltsége lesz a becslésünk. Ez egy alsó becslése a TSP optimumának, hiszen egy TSP megoldás is egy v_0-t nem tartalmazó feszítőfából, ami jelen esetben egy Hamilton-út és két élből áll, amikkel v_0 bekapcsolódik és ezeket becsültük alul az 1-Fa esetében. Ezekből tehát az eredeti c célfüggvény mellett fenáll:

$$TSP_{OPT}(c) \geq 1FA(c)$$

Most írjunk minden v csúcsra egy $\lambda_v \in \mathbb{R}$ számot (v_0-ra 0-t írunk) és vegyük a fent említett c' célfüggvényt az éleken. Ekkor egy optimális túra költsége minden v csúcsban $2{\cdot}\lambda_v$-vel nő. Felírva az 1-Fa becslést kapjuk, hogy:

$$TSP_{OPT}(c') = TSP_{OPT}(c) + 2\sum_{v \in V} \lambda_v \geq 1FA(c')$$

$$TSP_{OPT}(c) \geq 1FA(c') - 2\sum_{v \in V} \lambda_v$$

Az egyenlőtlenség jobb oldalán megjelenik a fent említett $L'(\lambda)$, amit Held-karp becslésnek nevezünk. Észrevehetjük, hogy ez egy erősebb becslés lesz, mint a fent látott $L(\lambda)$ függvény, hiszen eggyel kevesebb feltételt relaxáltunk. A feladatunk nyilván a minél jobb alsó becslés megtalálása, amit a következőképpen tudunk megfogalmazni:

$$L'_{max} = \max\left\{1FA(c') - 2\sum_{v \in V} \lambda_v : \lambda \in \mathbb{R}^n\right\}$$

Erre fogjuk használni az ún. szubgradiens módszert, ami lényegében szakaszonként lineáris, konkáv függggvéyek maximum keresésésével foglalkozik. Hasonlóan megfogalmazható konvex esetben is, de nekünk most erre lesz szükségünk. Látható, hogy $L'(\lambda)$ is ilyen. Konkáv, mert λ-ban lineáris függvények minimuma és mivel a lineáris függvények konkávak, ezeknek minimuma is az. Ráadásul mivel a feszítőfák, az élek és a csúcsok száma véges, így véges sok ilyen lineáris függvénynek a minimuma, tehát szakaszonként lineáris.

3.3. A szubgradiens módszer

A módszer ismertetése előtt kicsit foglalkozunk a szubgradiensek elméletével. A szubgradiens lényege az lesz, hogy kiterjesztjük az analízisből jól ismert derivált fogalmat nem differenciálható függvényekre is. Az f függvény deriváltja szemléletesen úgy értelmezhető egy $f : \mathbb{R}^n \mapsto \mathbb{R}$ adott x_0 pontjában, mint a függvény x_0 pontjába húzott érintő, ami egy olyan lineáris leképezés, amely legjobban közelíti f-t x_0-ban. Konkáv függvények esetében(ez az eset, amivel foglalkoznunk kell) a szubgradiens szintén érinti f-t az x_0 pontban, és minden más pontban nagyobb értéket vesz fel, mint f. Mivel egy adott pontban több ilyen is létezhet, ezért ezeknek halmazát nevezzük szubgradiensnek. A formális definició a következő:

3.3.1. Definíció:. *Vegyünk egy $f : \mathbb{R}^n \mapsto \mathbb{R}$ konkáv függvényt. Ekkor f-nek az x_0 pontban vett szubgradiense:*

$$\partial f(x_0) := \{v \in \mathbb{R}^n : f(x) \leq f(x_0) + \langle v, x - x_0 \rangle \quad \forall x \in \mathbb{R}^n\}$$

Most említsük meg pár nagyon fontos tulajdonságát a szubgradiensnek. Ezeket most bizonyítás nélkül közöljük.

3.3.2. Állítás. *A következőek teljesülnek:*

1. *Minden konkáv függvénynek létezik szubgradiense.*
2. *A függvény differenciálható az x_0 a pontban, akkor és csak akkor, ha a szubgradiens egyelemű és ez az elem $f'(x_0)$.*
3. *A 0 pontosan akkor szubgradiens, ha f-nek x_0-ban maximumhelye van.*

A szubgradiensre gondolhatunk úgy mint egy javító irányra. Ha a Held-Karp becslés során egy v csúcs fokszáma túl nagy, akkor a hozzátartozó λ_v-t növeljük, ezáltal a hozzákapcsolódó élek költsége is nő, így a következő 1-Fa becslésünk már kevesebb élét használja. Ugyanez elmondható, ha egy v csúcs fokszáma 1, akkor λ_v-t próbáljuk minél kevésbé növelni, hogy a becslésünk minél inkább használja a csúcsot.

Most már beszélhetünk magáról a módszerről, amit nem fogunk teljes lényegében megvizsgálni csak az alapgondolatával foglalkozunk.

Szubgradiens módszer

A módszer során lényegében mindig egy adott pontból indulunk, majd a pontbeli szubgradiens irányába ellépve bizonyos lépéshosszal eljutunk a következő pontukba, ahol a függvény értéke már nagyobb lesz. A kérdés, hogy el tudunk-e jutni a maximumhelyhez. Kicsit precízebben:

$$x^{i+1} = x^i + \theta^i v^i \tag{3.10}$$

Itt $x^i \in \mathbb{R}^n$ egy adott pont, $v^i \in \partial f(x^i)$ és $\theta^i \in \mathbb{R}$ az i-edik iterációhoz tartozó lépéshossz. A problémás része a módszernek a megfelelő lépéshossz megválasztása. Most erre nézünk egy lehetséges eljárást, ami a jelenlegi Held-Karp becslésünk során lesz hasznos.

Ha tudnánk előre L'_{max} értékét, akkor adott x^i pontból addig lépnénk a szubgradiens irányába, ameddig elérjük L'_{max}-ot, így érkezve az x^{i+1} pontba. Ezt a lépést, így tudjuk megfogalmazni:

$$L'_{max} = f(x^i) + (x^{i+1} - x^i)v^i$$

A (3.10) lépést használva a lépéshossz ekkor kifejezhető:

$$\theta^i := \frac{L'_{max} - f(x^i)}{||v^i||^2} \tag{3.11}$$

Természetesen általában nem vagyunk tisztában a maximumértékkel, így ezt egy az egyben nem tudjuk felhasználni. Vegyük észre, hogy a Held-Karp becslés egy felső becslését kapjuk, ha egy tetszőleges Hamilton-kör költségét vesszük. Az eljárásunk, tehát az lesz, hogy a becslésünk kiszámítása után mohón keresünk egy Hamilton-kört és annak a költségét használjuk a lépéshossz meghatározásához. Legyen H^i az i-edik iterációban talált Hamilton kör-költsége. Ekkor a lépéshosszunk:

$$\theta^i := \frac{H^i - f(x^i)}{||v^i||^2} \tag{3.12}$$

Előfordulhat az a kedvezőtlen eset, hogy túllépünk a maximumhelyen, ezért érdemes valamilyen β^i konstansokkal szabályozni az ellépés nagyságát. Ezeket úgy érdemes megválasztani, hogy az algoritmus elején legyenek viszonylag nagyok (pl. legyen $\beta^i = 2$), aztán ha egy idő után nem javítunk a módszer során, kezdjünk el kicsiket visszafelé lépegetni. Ha újra el kezdünk javítani, akkor maradjunk ugyanazon β értéknél. Bevezethetünk egy leállási feltételt is olyan értelemben, hogy az aktuális β már egy előre megadott ϵ korlát alá esik, akkor álljunk meg.

Az algoritmus konvergenciájáról általában semmit nem tudunk, mégis a gyakorlati haszna nagyon jelentős.

4. A PCSTP és a PTSP probléma

Ebben a fejezetben a fent említett PCSTP és PTSP problémákat matematikailag is megfogalmazzuk, illetve ismertetjük a Goemans-Williamson által megalkotott lényegében 2-approximációs algoritmust, amelyet majd a PTSP esetében kicsit módosítunk. A fejezet során elsősorban az [5] cikk gondolatmenetét fogjuk követni néhány helyen kiegészítve a [6] cikkben található ötletekkel.

4.1. A problémák leírása

4.1.1. Definíció: (PCSTP probléma). *Adott egy $G = (V, E)$ irányítatlan gráf az élein egy $c : E \mapsto \mathbb{Q}^+$ költségfüggvénnyel, illetve a csúcsain egy $\pi : V \mapsto \mathbb{Q}^+$ úgynevezett ,,büntetésfüggvénnyel". A csúcsok között van egy kitüntetett $r \in V$, amelyet gyökérnek nevezünk. A feladatunk találni egy olyan $T = (V_T, E_T)$ fát, amely tartalmazza r-et és minimalizálja a következő kifejezést:*

$$c(T) = \sum_{e \in E_T} c(e) + \sum_{v \in V \setminus V_T} \pi(v)$$

4.1.2. Definíció: (PTSP probléma). *Adott egy $G = (V, E)$ irányítatlan gráf az élein egy $c : E \mapsto \mathbb{Q}^+$ költségfüggvénnyel, amely teljesíti a háromszög-egyenlőtlenséget, illetve a csúcsain egy $\pi : V \mapsto \mathbb{Q}^+$ ún. ,,büntetésfüggvénnyel". A csúcsok között van egy kitüntetett $r \in V$, amelyet gyökérnek nevezünk. A feladatunk találni egy olyan $T = (V_T, E_T)$ túrát, amely tartalmazza r-et és minimalizálja a következő kifejezést:*

$$c(T) = \sum_{e \in E_T} c(e) + \sum_{v \in V \setminus V_T} \pi(v)$$

4.2. Goemans-Williamson primál-duál algoritmusa

Most rátérünk a Goemans-Willamson algoritmus ismertetésére, amire később majd csak GW-algoritmusként hivatkozunk. A PCSTP esetével kezdünk, ahol a feladatunk egy olyan Steiner-fát találni, amely tartalmazza a gyökérnek kinevezett csúcsot, illetve minimalizálja éleinek összköltségét, plusz a meg nem látogatott, azaz az általa ki nem feszített csúcsokért fizetett büntetést.

Először is megfogalmazzuk a problémánkat IP-feladatként. Minden élhez tartozni fog egy x_e bináris változó, amire $x_e = 1$ pontosan akkor, ha az

adott e élet belevesszük a megoldás Steiner-fába. Legyen $\mathcal{F}$ a gyökértől különböző csúcsoknak a hatványhalmaza.(Tehát a $V \setminus \{r\}$ csúcshalmaz összes részhalmazából álló halmazrendszer.) Az egészértékű programozási feladatban minden $\mathcal{F}$-ben lévő U halmazhoz felveszünk egy x_U változót. Minden ilyen halmaznak lesz egy büntetés szerinti költsége: $c_\pi(U) = \sum_{v \in U} \pi(v)$. Legyen $\delta(U)$ azon élek halmaza, amelyeknek pontosan egyik végpontja van egy adott $U \in \mathcal{F}$ halmazban, illetve legyen T a PCSTP megoldásunk élhalmaza. Legvégül legyen A azon U halmazok uniója, amelyek részhalmazai egy olyan U' halmaznak, amelyre $x_{U'} = 1$. Az ilyen U' halmazok olyanok lesznek, amelyekbe nem lép bele T-beli él. Vegyük észre, hogy minden $U \in \mathcal{F}$ halmaz esetében a következő két állítás közül valamelyik teljesül: $|\delta(U) \cap T| \geq 1$ vagy $U \subset A$, vagyis minden gyökeret nem tartalmazó csúcshalmazba belép egy T-beli él, ha viszont nem, akkor része egy olyannak, amibe nem. Ezek ismeretében tekintsük a következő IP-programot:

$$\sum_{e \in \delta(U)} x_e + \sum_{U' : U \subseteq U'} x_{U'} \geq 1 \quad \forall U \in \mathcal{F} \tag{4.1}$$

$$\min \sum_{e \in E} c(e) x_e + \sum_{U \in \mathcal{F}} c_\pi(U) x_U \tag{4.2}$$

$$x_e, x_U \in \{0, 1\} \quad \forall e \in E, \quad \forall U \in \mathcal{F}$$

Az IP-program (4.1) részében azt ellenőrizzük, hogy minden gyökért nem tartalmazó csúcshalmaz teljesíti-e a fent említett tulajdonságot, tehát vagy része egy olyan U' halmaznak, amire $x_{U'} = 1$, vagy van olyan él, amire $x_e = 1$ (tehát T-beli), és belép a halmazba. Megfigyelhető, hogy ez elég ahhoz, hogy egy általunk kívánt megoldást kapjunk, mert egyébként T nem lenne összefüggő vagy nem tartalmazná r-t, illetve a minimalizálás miatt a megoldás nem tartalmazhat kört, mert ekkor feleslegesen fizetnénk egy élért.

Most megkonstruáljuk az IP-programunk LP-relaxáltjának a duál programját. Ehhez felveszünk egy y_U változót minden $U \in \mathcal{F}$-hez. Most két feltételünk is lesz. Az egyik minden él esetében azon U halmazok duál értékeit korlátozza c-vel, amelyeknél az adott e élre $e \in \delta(U)$. A másik pedig minden U halmaz részhalmazainak duál értékeit korlátozza c_π-vel.

A duál feladat:

$$\sum_{U \in \mathcal{F} : e \in \delta(U)} y_U \leq c(e) \quad \forall e \in E \tag{4.3}$$

$$\sum_{U' : U' \subseteq U} y_{U'} \leq c_\pi(U) \quad \forall U \in \mathcal{F} \tag{4.4}$$

$$\max \sum_{U \in \mathcal{F}} y_U \tag{4.5}$$
$$y_U \geq 0 \quad \forall U \in \mathcal{F}$$

Most már rátérhetünk az algoritmus ismertetésére. Az alapötlet az lesz, hogy kiindulunk a primál feladat egy nem megengedett megoldásából, illetve a duál feladat egy megengedett megoldásából. Az algoritmus során megpróbáljuk a primált minél ,,megengedettebbé" tenni, a duál értékeket pedig addig növeljük majd, amíg lehetséges. Akkor leszünk készen, ha a duál értékeket már nem tudjuk növelni (4.3) vagy (4.4) megsértése nélkül. Ezt most precízebben is leírjuk.

Az egészre úgy tekintünk, mint egy idő alatt lejátszodó folyamat, vagyis x^τ és y^τ alatt azt fogjuk érteni, hogy egy adott $\tau \geq 0$ időpontban éppen mi az aktuális primál illetve duál megoldásunk. Kezdetben, tehát $\tau = 0$ esetén legyen $x^0 = 0$ és $y^0 = 0$ minden primál/duál változójában. Egy $e \in E$ élet **feszesnek** nevezzünk a τ időpontban, ha (4.3)-t egyenlőséggel teljesíti, azaz $y^\tau a_e = c(e)$, ahol a_e az e élhez tartozó oszlop az IP mátrixos felírásában. Ugyanez $U \in \mathcal{F}$ halmazokra is megfogalmazható csak ekkor a (4.4) teljesül egyenlőséggel, azaz $y^\tau a_U = c_\pi(U)$. Legyen F^τ azon élek halmaza x^τ-ra vonatkozólag, amelyek feszesek a τ időpontban. Kezdetben F^0 üres. Hasonlóképpen legyen $\mathcal{F}^\tau$ a feszes halmazok családja a τ időpontban x^τ-ra vonatkozólag.

Tegyük fel, hogy x^τ nem megengedett különben készen vagyunk. Legyen egy $U \in \mathcal{F}$ **aktív**, ha a következő három tulajdonságot teljesíti:

- U nem feszes
- U ki van feszítve F^τ valamelyik komponense által
- nincs olyan $e \in \delta(U)$ él, ami feszes

Az aktív halmazok esetében lényegében olyan összefüggő komponenseket fogunk érteni, amelyek még nem tartalmazzák a gyökeret. Legyen az aktív halmazok családja a τ időpontban $\mathcal{A}^\tau$. Látható, hogy $\mathcal{A}^0$ elemei kezdetben minden csúcs, hiszen ezek önmagukban egy összefüggő komponenst alkotnak. A GW algoritmus minden $\tau \geq 0$ időpontban megnöveli a duál értékeket egységesen minden $U \in \mathcal{A}^\tau$ halmazra. Ez az LP-relaxáltat vizsgálva a következőképpen történik. Szeretnénk y^τ-ból $y^{\tau+1}$-be lépni. A duál szimplex

módszerhez hasonlóan fogunk eljárni. Vesszük a τ időpontbeli aktív halmazok indikátorvektorát, tehát az aktív halmazok koordinátáiban 1-es szerepel a többinél 0. Legyen ez i_τ. Ekkor a lépésünk a következőképpen néz ki: $y^{\tau+1} = y^\tau + \epsilon i_\tau$ valami $\epsilon > 0$-ra. Az ϵ értékét kell már csak meghatároznunk, mégpedig úgy, hogy a duál feltételek ne sérüljenek. Tehát $y^{\tau+1}a_e \leq c(e) \quad \forall e \in E$ és $y^{\tau+1}a_U \leq c_\pi(U) \quad \forall U \in \mathcal{F}$. Ezen feltételekbe behelyettesítve $y^{\tau+1}$-t a következőt kapjuk:

$$\epsilon \leq \frac{c(e) - y^\tau a_e}{i_\tau a_e}, \quad \epsilon \leq \frac{c_\pi(U) - y^\tau a_U}{i_\tau a_U} \quad \forall e \in E, \quad \forall U \in \mathcal{F}$$

A legjobb ϵ választás a következő lesz:

$$\epsilon = \min\left(\min_{e \in E}\left(\frac{c(e) - y^\tau a_e}{i_\tau a_e}\right), \min_{U \in \mathcal{F}}\left(\frac{c_\pi(U) - y^\tau a_U}{i_\tau a_U}\right)\right)$$

Ha megtörtént a növelő lépés, akkor két esemény következhet be $y^{\tau+1}$-re vonatkozólag:

- Első esetben egy $e \in \delta(U)$ él feszessé válik valamilyen $U \in \mathcal{A}^\tau$-ra. Ekkor a hozzátartozó x_e változót 1-re állítjuk és hozzávesszük e-t $F^{\tau+1}$-hez. Ekkor lényegében az történik, hogy két összefüggő komponent összekötünk egy éllel és a legközelebbi iterációban már egy komponensként gondolunk rájuk. Ha a két komponens aktív volt, akkor a keletkező is az lesz, ha legalább az egyik deaktivált, olyan értelemben, hogy tartalmazza r-t, akkor a keletkező is deaktiválódik. Megemlítendő, hogy két deaktivált komponenst nem húzunk össze éllel, mert feleslegesen fizetnénk érte.

- Második esetben valamelyik $U \in \mathcal{A}^\tau$ válik feszessé. Ekkor a hozzátartozó x_u változót állítjuk be 1-re. Ekkor az U halmaz deaktíválódik olyan értelemben, hogy a megoldásunk kihagyja az elemeit, mert jobban megéri büntetést fizetni értük. Ekkor az algoritmus U minden csúcsát megjelöli mégpedig a komponens nevével.

A két esemény egyébként egy iterációban is megtörténhet. Mindkét esetben frissítjük az aktív halmazokat és folytatjuk a duál változók növelését addig, amíg tudjuk. Ez akkor következik be, ha $\lambda = 0$, ami azt is jelenti, hogy már nincsen aktív halmaz. Ezután megkeressük azon U halmazokat, amelyekre $x_U = 1$, és minden csúcsa elérhető a gyökérből feszes éleken keresztül az

aktuális megoldásunkban. Ekkor x_U-t 0-ra állítjuk ezen halmazok esetében, hiszen őket nem fogja kihagyni a megoldásunk, így nem is kell értük büntetést fizetni.

Az algoritmus azzal zárul, hogy el kezdünk mohón éleket törölni a gráfból úgy, hogy két feltételt megőrizzünk. Először is minden meg nem jelölt csúcsot el tudjunk érni a gyökérből, hiszen ezek nem voltak benne deaktivált komponensben az algoritmus során és az algoritmus sosem volt hajlandó büntetést fizetni értük. Másodszor, ha egy C-vel jelölt csúcs elérhető a gyökérből, akkor minden olyan csúcs is, aminek a jelölése C' és $C \subseteq C'$ Ezzel megkapunk egy r-t tartalmazó T fát illetve a kihagyott csúcsok uniójaként előálló A halmazt.

Ekkor a megoldásunk költsége: $c_{GW}(T, A) = \sum_{e \in T} c(e) + \sum_{v \in A} \pi(v)$

4.2.1. Tétel. *Tegyük fel, hogy a GW algoritmus outputjként megkapunk egy T fát, illetve egy A csúcshalmazt és egy $\{y_U\}_{U \in \mathcal{F}}$ megengedett duál megoldást. Ekkor a következőek teljesülnek: a (T, A) páros megengedett primál megoldás és*

$$c_{GW}(T, A) \leq \left(2 - \frac{1}{n-1}\right) \sum_{U \in \mathcal{F}} y_U \leq \left(2 - \frac{1}{n-1}\right) c_{opt}$$

itt c_{opt} a PCSTP optimuma.

Azaz a GW-algoritmus $2 - \frac{1}{n-1}$ approximációs, tehát kisebb csúcsszám esetén egyre jobb közelítést kapunk.

Bizonyítás. A bizonyítást a [6] cikk alapján fogjuk közölni. A második egyenlőtlenség nyilván a gyenge dualitás-tétel miatt teljesül, tehát a tétel ezen része triviálsinak mondható.

Először is kezdejük azzal, hogy a GW algoritmus tényleg megendedett megoldást szolgáltat. Ez nyilvánvaló, hiszen minden halmaz deaktivált a végen, ami azt jelenti, hogy vagy van a belépő élei között T-beli él is (ekkor r-ből elérhető) vagy egyik csúcsa sem érhető el r-ből. Ez pontosan az a feltétel, amit az elején megköveteltünk, így a megoldásunk megengedett.

Most térjünk rá az approximáció bizonyítására. Minden T által le nem fedett csúcs az algoritmus során deaktivált komponensek valamelyikében fekszik. Ráadásul, ha egy csúcs egy C deaktivált komponensben volt, akkor C többi csúcsát sem érhetjük el T által. Ez következik abból a feltételből, amit az élek törlésénél feltettünk. Ezek fényében az A csúcshalmaz felbontható $C_1, \ldots C_k$ diszjunkt deaktivált komponensekre. Ezek a halmazok maximálisak azon halmazok közül, amellyel megjelöltünk egy csúcsot az algoritmus során. Mivel minden C_j egy deaktivált komponens, azon belül is

feszes, ezért $\sum_{U\subseteq C_j} y_U = \sum_{v\in C_j} \pi(v)$. Felhasználva ezt a bizonyítandó állítás átírható a következő alakra: $\sum_{e\in T} c(e)+\sum_j \sum_{U\subseteq C_j} y_U \leq \left(2-\frac{1}{n-1}\right)\sum_{U\in\mathcal{F}} y_U$. Minden $e \in T$ él feszes, tehát egyenlőséggel teljesül a hozzájuk tartozó feltétel: $\sum_{U:e\in\delta(U)} y_U = c(e)$. A következőt kell bizonyítanunk:

$$\sum_{e\in T}\sum_{U:e\in\delta(U)} y_U + \sum_j \sum_{U\subseteq C_j} y_U \leq \left(2-\frac{1}{n-1}\right)\sum_{U\in\mathcal{F}} y_U$$

Ez átírható a következőképpen:

$$\sum_{U\in\mathcal{F}} y_U|\delta(U)\cap T| + \sum_j \sum_{U\subseteq C_j} y_U \leq \left(2-\frac{1}{n-1}\right)\sum_{U\in\mathcal{F}} y_U$$

A tételt τ szerinti indukcióval fogjuk bizonyítani, azaz a fő ciklus szerint. Vegyünk egy tetszőleges iterációt. Ehhez megalkotjuk a következő $H(V', E')$ gráfot: a csúcsok legyenek a τ időpontbeli aktív és deaktivált komponensek, az élek pedig minden C aktív komponens esetében legyenek olyanok, amelyek szerepelnek a T megoldásunkban és belépnek az adott C komponensünkbe, tehát formálisan $E' := \{e \in E : e \in \delta(C) \cap T\}$, ahol C egy aktív komponens a τ időpontban. Az izolált, inaktív csúcsokat elhagyhatjuk, ezekre nem lesz szükségünk. Legyen N_a az aktív csúcsok, N_i pedig az inaktív csúcsok halmaza. Végül legyen N_d azon aktív csúcsok halmaza, amelyek az algoritmus során egyszer deaktiválódnak olyan értelemben, hogy a csúcsait megjelöljük a komponens nevével. Jól látható, hogy ezek a csúcsok H-ban izoláltak, hiszen az algoritmus végén minden bemenő élét töröljük egy ilyen komponensnek, mivel a feltételeket ezzel nem sértjük meg. Az iterációban történő növelés után a bizonyítandó egyenlőtlenség jobb oldala $\epsilon\left(2-\frac{1}{n-1}\right)|N_a|$-val változik, illetve a bal oldala pedig $\epsilon\left(\sum_{v\in N_a} d(v) + |N_d|\right)$-vel, ahol $d(v)$ a v csúcs H-ban lévő fokszámát jelöli. Ezek a változások egyszerűen következnek abból, hogy hogyan lépünk y^τ-ból, $y^{\tau+1}$-be. Az indukció miatt elég belátnunk azt, hogy $\sum_{v\in N_a} d(v) + |N_d| \leq \left(2-\frac{1}{n-1}\right)|N_a|$, hiszen ekkor az egyenlőtlenség az algoritmus során végig fennáll. Mivel minden N_d-ben levő csúcs foka 0, ezért a következő összefüggést elég lesz belátnunk:

$$\sum_{v\in N_a - N_d} d(v) \leq \left(2-\frac{1}{n-1}\right)|N_a - N_d| \tag{4.6}$$

Ez egy erősebb állítás mint az előző, de a k-MST esetében pont ezt fogjuk kihasználni.

Szükségünk lesz arra, hogy H egy fa minden τ időpontban leszámítva az aktív izolált csúcsokat. Ez persze $\tau = 0$ esetén nyilvánvaló, hiszen akkor H csúcsai pontosan az eredeti csúcsok, az élek pedig T élei, azaz H azonos a megoldásfával. Nézzük meg mi történhet H-val egy adott τ és $\tau + 1$ időpont között. Ha az algoritmusban egy T-beli élt veszünk fel két összefüggő komponens között, akkor azt az élet H-ban összehúzzuk vagyis a két végpontja helyett felveszünk egy új csúcsot, amit a két végpont összes szomszédjával összekötünk. Egy ilyen művelet a fatulajdonságot nem rontja el. Ha egy nem T-beli élet veszünk fel, akkor azt fogjuk látni, hogy ez H-n lényegében nem változtat. Az ilyen élek a végén törlésre kerülnek, azaz olyan komponenseket kapcsolnak össze, amelyeknél legalább eg y esetében kitöröljük az összes élét. Ezek H-ban vagy két izolált csúcshoz tartozó komponenst kötnének össze (ezek lehetnek aktívak és inaktívak is), vagy a H-beli fa egy csúcsát kötné össze egy izolált csúccsal. Az él felvétele után vagy eltűnik egy izolált csúcs vagy kettő helyett egy újat veszünk fel. Nyilván H-nak a fatulajdonságán ezek nem változtatnak. Egy komponens deaktiválása pedig nem változat H-n, hiszen a másik végpontja még aktív. Akkor lehetne baj ha esetleg egy H-beli él két végpontját deaktiválnák egymás után, de ez megoldható úgy, hogy a hozzájuk tartozó két komponensből először csinálunk egy nagyobbat az adott él felvételével, majd utána deaktiváljuk a nagyobb komponenst. Összefoglalva bármilyen átalakítást végzünk, H egy fa lesz minden τ időpontban.

Most belátjuk, hogy H-ban legfeljebb egy deaktivált komponenshez tartozó levél lehet, azaz minden más levele H-nak aktív csúcs. Tegyük fel, hogy van H-nak egy olyan deaktivált v csúcsa, ami egy megjelölt C_v komponenshez tartozik és az egyetlen kimenő éle e, tehát C_v nem csatlakozik a gyökérhez, így minden csúcsa meg van jelölve C_v-vel. Továbbá mivel v levél, ezért őt nem használhatjuk fel meg nem jelölt csúcs elérésére, azaz nem lehet egy olyan úton rajta, ami a gyökérből indul és egy jelöletlen csúcsba tart. A T megoldásunk szerkesztéséből pedig az következik, hogy $e \notin T$, ami ellentmondás. A H gráfunk maximum egy inaktív levelet tartalmazhat, ami a gyökeret tartalmazó komponensre utal.

Ezek után:

$$\begin{aligned}\sum_{v\in N_a - N_d} d(v) &\leq \sum_{v\in (N_a-N_d)\cup N_i} d(v) - \sum_{v\in N_i} d(v)\\ &\leq 2(|(N_a - N_d)\cup N_i| - 1) - (2|N_i| - 1)\\ &= 2|N_a - N_d| - 1\\ &\leq \left(2 - \frac{1}{n-1}\right)|N_a - N_d|\end{aligned}$$

Az első egyenlőtlenség esetében felhasználjuk, hogy minden H-ban kifeszített dekativált csúcsnak legalább 2 a foka egy kivételével. Az utolsó egyenlőtlenség esetében pedig azt használjuk ki, hogy az aktív csúcsok száma legfeljebb $n-1$. Ezzel a tételt be is láttuk □

A GW algoritmust használhatjuk a PTSP probléma megoldására is egy kis módosítással. Először lefutattjuk az algoritmust a gráfon, viszont a büntetéseket elfelezzük minden csúcsra. Legyen $\pi' := \frac{\pi}{2}$ az új büntetésfüggvény. Az algoritmus outputjaként kapott fából úgy csinálunk túrát, hogy megduplázunk minden élt a fában ugyanazon élköltséggel. Ekkor találunk egy Euler-körsétát a fa csúcsain (minden csúcs foka páros). Ha a séta folyamán ismétlődő csúcshoz érkezünk, hagyjuk el és a sorban következő csúcsot kössük hozzá az előzőhöz, ha az még nem szerepelt. A minimalitáson ez nem változtat a háromszög-egyenlőtlenség miatt. Az algoritmus $2 - \frac{1}{n-1}$ approximációs tulajdonságához szükségünk arra, hogy IP-feladatként is megfogalmazzuk a problémát:

$$\sum_{e\in\delta(U)} x_e + \sum_{U':U\subseteq U'} x_{U'} \geq 2 \quad \forall U \in \mathcal{F} \tag{4.7}$$

$$\min \sum_{e\in E} c(e)x_e + \sum_{U\in\mathcal{F}} c_{\pi'}(U)x_U \tag{4.8}$$

$$x_e \in \{0,1\} \quad \forall e \in E, \quad x_U \in \{0,2\} \quad \forall U \in \mathcal{F}$$

A (4.7) feltétel foglalkozik azzal, hogy a kiválasztott élhalmazunk egy túrát alkosson, ami a kihagyott csúcsokat nem érinti. Ez akkor lehetséges, ha minden csúcshalmazt legalább kétszer érint a túra vagy teljesen kihagy és ekkor minden csúcsáért büntetést fizetünk. Mivel a kihagyott halmazok esetében $x_U = 2$ és $c_{\pi'}(U) = \frac{c_\pi(U)}{2}$, így az eredeti büntetést fizetjük minden halmazon.

Most tekintsük a primál feladat LP-relaxáltjának a duál feladatát:

$$\sum_{U\in\mathcal{F}:e\in\delta(U)} y_U \leq c(e) \quad \forall e \in E \tag{4.9}$$

$$\sum_{U':U'\subseteq U} y_{U'} \leq c_{\pi'}(U) \quad \forall U \in \mathcal{F} \tag{4.10}$$

$$\max 2 \cdot \sum_{U\in\mathcal{F}} y_U \tag{4.11}$$

$$y_U \geq 0 \quad \forall U \in \mathcal{F}$$

Ezekből már megfogalmazhatunk egy tételt.

4.2.2. Tétel. *Ezen módosítása a GW-algoritmusnak* $2-\frac{1}{n-1}$*-approximációs, azaz:*

$$c_{GW}(C,A) \leq \left(2-\frac{1}{n-1}\right)\cdot c_{opt},$$

ahol C *az algoritmus által adott túra és* c_{opt} *a PTSP feladat optimuma.*

Bizonyítás.

$$c_{GW}(C,A) \leq 2{\cdot}c_{GW}(T,A) \leq \left(2-\frac{1}{n-1}\right){\cdot}\left(2\cdot\sum_{U\in\mathcal{F}} y_U\right) \leq \left(2-\frac{1}{n-1}\right){\cdot}c_{opt}$$

Itt T a GW-algoritmus által adott Steiner-fa. Az első egyenlőtlenség azért teljesül, mert a kapott Euler-körsétában még javítunk, így a túra költsége nem nőhet. A második az előző tételből következik. A harmadik meg azért, mert most a duál optimumunk a $2{\cdot}\sum_{U\in\mathcal{F}} y_U$, ami a gyenge dualitás tétel miatt becsülhető bármilyen primál megoldás költségével, így az egész optimummal is. □

5. A k-MST és a k-TSP probléma

Ebben a fejezetben a fent már említett k-MST és k-TSP problémákkal foglalkozunk. Leírjuk őket a matematika nyelvén majd ismertetjük Garg 5-approximációs algoritmusát. Ez a rész lényegében a [7] cikk gondolatmenetét követi, de néhány helyen (pl. jelölések során), azért használjuk a [5] cikket is.

5.1. A problémák megfogalmazása és modellezése

5.1.1. Definíció: (k-MST)**.** *Adott egy $G = (V, E)$ irányítatlan gráf az élein egy $c : E \mapsto \mathbb{Q}^+$ költségfüggvény, ami teljesíti a háromszög-egyenlőtlenséget, illetve egy k pozitív egész szám. A feladatunk találni egy olyan fát, ami pontosan k csúcsot feszít ki, illetve ezen belül az éleinek összköltsége minimális. Ha az elején adott egy $r \in V$ gyökércsúcs, akkor olyan fát kell találni, ami tartalmazza r-t.*

5.1.2. Definíció: (k-TSP)**.** *Adott egy $G = (V, E)$ irányítatlan gráf az élein egy $c : E \mapsto \mathbb{Q}^+$ költségfüggvény, ami teljesíti a háromszög-egyenlőtlenséget, illetve egy k pozitív egész szám. A feladatunk találni egy olyan túrát, ami pontosan k csúcsot tartalmaz, illetve ezen belül az éleinek összköltsége minimális. Ha az elején adott egy $r \in V$ gyökércsúcs, akkor olyan túrát kell találni, ami tartalmazza r-t.*

Először természetesen a k-MST problémával foglalkozunk. Most tekintsük a következő IP-programot, ami a feladatunkat írja le abban az esetben, amikor adott egy r gyökércsúcs is:

$$\sum_{e \in \delta(U)} x_e + \sum_{U' : U \subseteq U'} x_{U'} \geq 1 \quad \forall U \in \mathcal{F} \tag{5.1}$$

$$\sum_{U \in \mathcal{F}} x_U |U| \leq n - k \tag{5.2}$$

$$\min \sum_{e \in E} c(e) x_e \tag{5.3}$$

$$x_e, x_U \in \{0, 1\} \quad \forall e \in E, \quad \forall U \in \mathcal{F}$$

A jelölések ugyanazok mint az előző feladat esetében. Nyilván $n = |V|$. Az x_e bináris változó (hasonlóan az előzőhöz) 1, ha az adott e él bekerül

a megoldásunkba különben legyen 0. Az x_U bináris változó pedig akkor 1, ha az adott U halmazba nem lép T-beli él és nincs egy ugyanilyen tulajdonságú halmaz, ami őt tartalmazza, különben 0. A (5.1)-ben szereplő feltétel ugyanazt követeli meg, mint, amit a PCSTP esetében láttunk. A plusz feltételünk a (5.2) feltétel, ami arról gondoskodik, hogy a megoldásunk legalább k csúcsot tartalmazzon. Természetesen ennek az LP-relaxáltját fogjuk használni elsősorban, ahol nem követeljük meg a vátozóktól, hogy egészek legyenek. Most tekintsünk a Lagrange-relaxáltját a problémának, mégpedig úgy, hogy a (5.2) feltételt relaxáljuk. Az így kapott IP-programunk nagyon hasonlít a PCSTP esetében tárgyalt IP-programhoz, szóval az ötletünk majd az lesz, hogy a fent tárgyalt GW-algoritmust fogjuk majd meghívni szubrutinként. Tekintsük az így kapott IP-programot:

$$\sum_{e\in\delta(U)} x_e + \sum_{U':U\subseteq U'} x_{U'} \geq 1 \quad \forall U \in \mathcal{F}$$

$$\min \sum_{e\in E} c(e)x_e + \lambda \left(\sum_{U\in\mathcal{F}} x_U|U| - (n-k) \right) \tag{5.4}$$

$$x_e, x_U \geq 0 \quad \forall e \in E, \quad \forall U \in \mathcal{F}$$

Itt természetesen $\lambda \geq 0$, hiszen az eredeti feltételben egyenlőtlenség szerepelt. A minimalizásban a -$\lambda(n-k)$-s tag külön vehető, ugyanis ez nem függ a primál megoldástól. Az így megmaradt minimalizálandó részünk:

$$\sum_{e\in E} c(e)x_e + \lambda \sum_{U\in\mathcal{F}} x_U|U|$$

Látható, hogy az így kapott IP-program a PCSTP egyik változata, ahol $\pi(v) = \lambda$ minden v csúcsra. A Lagrange-relaxált tulajdonságaiból következik, hogy bármilyen megengedett megoldását vesszük a k-MST problémának az megengedett megoldása lesz a Lagrange-relaxáltjának, illetve a k-MST probléma optimumát alulról becsli (5.4).

Mielőtt még rátérünk Garg algoritmusára szükségünk lesz egy kis előkészületre, illetve a GW algoritmus felelevenítésére. A PCSTP problémát megoldó GW algoritmus outputjaként megkapunk egy T fát, ami tartalmazza r-t, illetve egy A halmazt, ami azon csúcsokból áll, amelyeket T nem fed le. Az algoritmus létrehoz egy y megengedett duál megoldást. Nézzük most a következő tételt ezekkel kapcsolatban, amire majd később szükségünk lesz.

5.1.3. Tétel. *Legyen (T,A) a GW algoritmus outputja, illetve y az általa alkotott duál megengedett megoldás. Ekkor:*

$$\sum_{e\in T} c(e) + \left(2 - \frac{1}{n-1}\right)\sum_{v\in A}\pi(v) \leq \left(2 - \frac{1}{n-1}\right)\sum_{U\in\mathcal{F}} y_U$$

Bizonyítás. Emlékezzünk vissza a GW algoritmus $2 - \frac{1}{n-1}$ közelítésének bizonyítására. Ott lényegében átalakítottuk az egyenlőtlenségünket, majd megnéztük, hogy egy iteráció után hogy változik meg a két oldal. Ehhez beláttuk a (4.6) egyenlőtlenséget, ami egy erősebb állítás volt mint, amit akkor bizonítani akartunk. Ezt a plusz információt felhasználhatjuk ezen tétel bizonyítására. Lényegében ugyanúgy kell eljárnunk mint az előző bizonyítás során. Ennek átgondolását az olvasóra bízzuk. □

Szükségünk lesz a tétel egy következményére, ami egyszerű átalakításokkal jön ki.

5.1.4. Következmény. *Ha $\pi(v) = \lambda$ valamilyen $\lambda \geq 0$-ra, akkor:*

$$\sum_{e\in T} c(e) + 2\lambda|A| \leq 2\sum_{U\in\mathcal{F}} y_U$$

Tekintsük most a duálját a k-MST LP Lagrange-relaxáltjának:

$$\sum_{U\in\mathcal{F}: e\in\delta(U)} y_U \leq c(e) \quad \forall e \in E \tag{5.5}$$

$$\sum_{U':U'\subseteq U} y_{U'} \leq |U|\lambda \quad \forall U \in \mathcal{F} \tag{5.6}$$

$$\max \sum_{U\in\mathcal{F}} y_U - \lambda(n-k) \tag{5.7}$$

$$y_U \geq 0 \quad \forall U \in \mathcal{F}$$

Most a 5.1.4 következmény mindkét oldalából vonjuk ki $2\lambda(n-k)$-t. Ekkor a következő egyenlőtlenség sorozat teljesül:

$$\sum_{e\in T} c(e) + 2\lambda(|A| - (n-k)) \leq 2\left(\sum_{U\in\mathcal{F}} y_U - \lambda(n-k)\right) \leq 2c_{opt} \tag{5.8}$$

Itt c_{opt} a k-MST probléma optimumát jelöli. A második egyenlőtlenség, abból következik, hogy a zárójelben szereplő tagunk az pont a duál Lagrange-relaxált optimuma, ami a gyenge dualitás-tétel miatt becsülhető a primál Lagrange-optimummal, ami pedig alsó becslése a primál LP-relaxált optimumának, ami meg persze alsó becslése a k-MST probléma optimumának.

Ebből az egyenőtlenségből azt kapjuk meg, hogy ha $\pi(v) = \lambda \quad \forall v \in V$ mellett futtatjuk a GW-algoritmust, és a végén $|A| = n - k$, akkor az így kapott outputunk költége nem lehet nagyobb az optimum kétszeresénél, tehát ekkor 2-approximációt érünk el. Persze ez egy nagyon szerencsés helyzet, így a többi esetet máshogy kell kezelnünk. Ha például több csúcsot hagy ki a GW-algoritmus mint $n - k$, akkor a megoldásunk ráadásul nem is megengedett, ellenkező esetben pedig semmit nem tudunk az approximácó nagyságáról. Megfelelő λ választásával, viszont elkerülhetőek ezek a problémák. Sajnos az approximációs számot egy kicsit meg kell majd növelnünk. Garg algoritmusa a megfelelő λ megtalálására fog törekedni, amit a következő részben részletesebben tárgyalunk.

5.2. Garg 5-approximációs algoritmusa

Mielőtt rátérünk az algoritmus ismertetésére szükségünk lesz két feltevésre, amit az algoritmus során felhasználunk majd. Először legyen $c_0 \leq c_{opt}$, ahol c_0 a legkisebb nemnegatív él költségét jelenti. Ezt nyilván feltehetjük, mert ellenkező esetben $c_{opt} = 0$. Ez ellenőrízhető az előkészületeknél, hiszen csak azt kell megnéznük, hogy van-e csak 0-költségű élekből álló és r-t tartalmazó k méretű fa. Másodszor bármely csúcs távolsága a gyökértől felül becsülhető az optimum költéségével. Ezt azért tehetjük fel, mert, ha egy csúcs távolabb van a gyökértől, mint az optimum, akkor ő nyilván nem kerülhet bele a megoldásunkba, így ezeket elhagyhatjuk a gráfból.

Az algoritmus úgy épül fel, hogy meghívja a GW-algoritmust szubrutinként adott λ esetén, majd annak fényében változatja az aktuális λ értékét. Vegyük észre, hogy $\lambda = 0$ esetében futtatva a GW-algoritmust az outputunk csak a gyökércsúcsból áll, hiszen ekkor nem fizetünk büntetést a kihagyott csúcsokért. Azt is megfigyelhetjük, ha $\lambda = \sum_{e \in E} c(e)$ esetében tesszük, akkor outputként egy összes csúcsot kifeszítő fát kapunk, hiszen ekkor lesz a lehető legkisebb költségű a megoldásunk, ha minden halmazt kifeszítünk. Ezekből megállapítható, hogy a lehetséges λ értékeket a $[0, \sum_{e \in E} c(e)]$ intervallumból vesszük. Garg algoritmusa lényegében egy bináris keresést végez. Általános lépésben vesszük a $[\lambda_1, \lambda_2]$ intervallumot adott λ_1, λ_2 értékek esetén. Kez-

detben $\lambda_1 = 0$ és $\lambda_2 = \sum_{e \in E} c(e)$. Minden lépésben legyen $\lambda = \frac{\lambda_1 + \lambda_2}{2}$ majd ezzel a λ-val futtatjuk a GW-algoritmust. Ha az így kapott fánk kevesebb, mint k csúcsból áll, akkor frissítsük λ_1-t λ-val. Ha több csúcsunk van mint k értelemszerűen λ_2-t frissítjük λ-val. Egyébként ha szerencsések vagyunk egy megengedett megoldást kapunk, ami nem rosszabb, mint az optimum kétszerese, ilyenkor megállhatunk. A keresést addig folytatjuk, ameddig a következő két feltétel egyszerre nem teljesül λ_1, λ_2-re:

1. $\lambda_2 - \lambda_1 \leq \frac{c_0}{2n(2n+1)}$ (c_0-t a fejezet elején már definiáltuk)
2. $i = 1, 2$ esetében futtatva az algoritmust $\lambda = \lambda_i$ választással kapunk egy (T_i, A_i) primál megoldást, ami k_i csúcsot feszít ki és egy $y^{(i)}$ duál megoldást és $k_1 < k < k_2$

Ezzekel garantáljuk a szubrutin hívások számának végességét, amit becsülni is tudunk.

5.2.1. Állítás. *Garg algoritmusa* $O\left(\log \frac{n^2 \sum_{e \in E} c(e)}{c_0}\right)$ *szubrutin hívást használ.*

Bizonyítás. Legyen N a hívások száma. Lényegében az a kérdés, hogy hányszor felezhető el a λ_1, λ_2 intervallum úgy, hogy az első feltétel még ne teljesüljön. Tehát mivel kezdetben $\lambda_1 = 0$ és $\lambda_2 = \sum_{e \in E} c(e)$:

$$\frac{c_0}{2n(2n+1)} \leq \frac{\sum_{e \in E} c(e)}{2^N}$$

Átalakítva:

$$N \leq \log \frac{2n(2n+1) \sum_{e \in E} c(e)}{c_0}$$

Ebből már jól látszik az állítás. □

Innentől fogva feltesszük, hogy nem sikerült k csúcsú fát találnunk. Az algoritmusunk ezen szakaszában megpróbáljuk a megkapott $(T_i, A_i) \quad i = 1, 2$ primál megoldásokból kihozni a megoldásunkat, ami már k csúcsot tartalmaz. Előtte viszont szükségünk lesz a 5.1.3 tétel egy következményére, amit egyszerű átrendezésekkel kapunk.

5.2.2. Következmény. *A bináris keresés után kapott* $i = 1, 2$ *esetén* (T_i, A_i) *primál és* $y^{(i)}$ *duál megoldásokra az alábbiak teljesülnek:*

$$\sum_{e \in T_1} c(e) \leq \left(2 - \frac{1}{n}\right) \left(\sum_{U \in \mathcal{F}} y_U^{(1)} - |A_1| \lambda_1\right) \tag{5.9}$$

$$\sum_{e\in T_2} c(e) \leq \left(2-\frac{1}{n}\right)\left(\sum_{U\in\mathcal{F}} y_U^{(2)} - |A_2|\lambda_2\right) \tag{5.10}$$

Most szeretnénk ezen két egyenlőtlenségnek a konvex kombinációját venni, amiből egy becslést kaphatunk T_1 és T_2 költségére az optimum függvényében. Ehhez keresünk α_1 és α_2 értékeket a következő tulajdonságokkal:

- $\alpha_1, \alpha_2 \geq 0$
- $\alpha_1 + \alpha_2 = 1$
- $\alpha_1|A_1| + \alpha_2|A_2| = n - k$

Legyen még minden $U \in \mathcal{F}$ esetén $y_U = \alpha_1 y_U^{(1)} + \alpha_2 y_U^{(2)}$. A feltételekből egy egyenletrendszer megoldásával belátható, hogy:

$$\alpha_1 = \frac{n-k-|A_2|}{|A_1|-|A_2|} \quad \alpha_2 = \frac{|A_1|-(n-k)}{|A_1|-|A_2|}$$

Most már kimondhatunk egy lemmát, amit majd később felhasználunk az approximáció bizonyításához.

5.2.3. Lemma.
$$\alpha_1 \sum_{e\in T_1} c(e) + \alpha_2 \sum_{e\in T_2} c(e) < 2c_{opt}$$

Bizonyítás.

$$\begin{aligned}
\sum_{e\in T_1} c(e) &\leq \left(2-\frac{1}{n}\right)\left(\sum_{U\in\mathcal{F}} y_U^{(1)} - |A_1|(\lambda_1+\lambda_2-\lambda_2)\right) \\
&\leq \left(2-\frac{1}{n}\right)\left(\sum_{U\in\mathcal{F}} y_U^{(1)} - |A_1|\lambda_2\right) + \left(2-\frac{1}{n}\right)\frac{c_0}{2n(2n+1)} \\
&< \left(2-\frac{1}{n}\right)\left(\sum_{U\in\mathcal{F}} y_U^{(1)} - |A_1|\lambda_2\right) + \frac{c_0}{2n+1}
\end{aligned}$$

Itt a (5.9) egyenlőtlenségbe becsempésztünk egy λ_2-t, majd használtuk a leállási feltételünket. A konvex kombinációra alkalmazva a (5.10) egyenlőtlenséget és az eddigi becslésünket:

$$\alpha_1 \sum_{e\in T_1} c(e) + \alpha_2 \sum_{e\in T_2} c(e) <$$

$$
\begin{aligned}
&< \alpha_1 \left(2 - \frac{1}{n}\right) \left(\sum_{U \in \mathcal{F}} y_U^{(1)} - |A_1|\lambda_2\right) + \frac{\alpha_1 c_0}{2n+1} \\
&\quad + \quad \alpha_2 \left(2 - \frac{1}{n}\right) \left(\sum_{U \in \mathcal{F}} y_U^{(2)} - |A_2|\lambda_2\right) \\
&= \left(2 - \frac{1}{n}\right) \left(\sum_{U \in \mathcal{F}} (\alpha_1 y_U^{(1)} + \alpha_2 y_U^{(2)}) - (\alpha_2|A_2| + \alpha_1|A_1|)\lambda_2\right) + \frac{\alpha_1 c_0}{2n+1} \\
&= \left(2 - \frac{1}{n}\right) \left(\sum_{U \in \mathcal{F}} y_U - (n-k)\lambda_2\right) + \frac{\alpha_1 c_0}{2n+1} \qquad (5.11) \\
&\leq \left(2 - \frac{1}{n}\right) c_{opt} + \frac{\alpha_1 c_0}{2n+1} \qquad (5.12) \\
&\leq \left(2 - \frac{1}{n}\right) c_{opt} + \frac{1}{2n+1} c_{opt} \qquad (5.13) \\
&\leq 2c_{opt}
\end{aligned}
$$

A (5.11)-es egyenlőség α_1, α_2 tulajdonságaiból következik. (5.12)-nél használjuk a fent említett 5.8 egyenlőtlenséget. Végül (5.13) esetében a feltevésünket alkalmazzuk a legkisebb költségű nemnegatív élre vonatkozólag és azt, hogy $\alpha_1 \leq 1$. □

Ettől a ponttól kezdve Garg algoritmusa két irányba mehet. Első esetben tegyük fel, hogy $\alpha_2 \geq \frac{1}{2}$. Ekkor mivel T_2 megengedett megoldás (hiszen több mint k csúcsot tartalmaz), válasszuk T_2-t a megoldásunknak. A 5.2.3 lemma segítségével belátható, hogy ekkor az optimum négyszeresénél nem kaphatunk rosszabb megoldást.

$$\sum_{e \in T_2} c(e) \leq 2\alpha_2 \sum_{e \in T_2} c(e) \leq 4c_{opt}$$

Ekkor persze nyugodtan elhagyhatunk leveleket T_2-ből, hogy pontosan k csúcsú fánk legyen, hiszen közben a 4-approximáció megmarad. Másik esetben tegyük fel, hogy $\alpha_2 < \frac{1}{2}$. Most úgy fogunk megoldást csinálni, hogy fogjuk T_1-et és kipótoljuk T_2-beli csúcsokkal így egy k csúcsú fához jutunk. Először is vegyük azon csúcsokat, amelyek nincsenek benne T_1-ben, de T_2-ben igen. Legyen ezek száma $l \geq k_2 - k_1$. A PTSP-hez hasonlóan megduplázzuk T_2 éleit és az így kapott Euler-sétát nemcsak csúcsismétlődés

esetén rövidítjük hanem, ha olyan csúcshoz jutunk, amelyik benne van T_1-ben. Ezek után egy l hosszú túrát kapunk T_2-ben. Ebben a túrában keressük meg a legrövidebb $k-k_1$ csúcsból álló utat. Ezek mind mennek $O(n)$ időben. Végül ezt a $k-k_1$ hosszú utat kell hozzákapcsolnunk a gyökérhez valamelyik éllel (természetesen érdemes a legkisebb költségűvel) és ekkor kész a megoldásunk. Most már csak az approximáció bizonyítása van hátra. Szeretnénk a gyökérhez kapcsolandó utunk költségét felülről becsülni. Ehhez adjuk össze a lehetséges $k-k_1$ hosszú utak költségeit. Legyen ez az összeg S, amiben lényegében minden $T_2 \setminus T_1$-beli él költsége $k-k_1$-szer szerepel, hiszen egy él ennyi útban szerepelhet. Ekkor a következő teljesül:

$$(k_2-k_1)c_{min} \leq S = (k-k_1)\cdot 2\sum_{e\in T_2\setminus T_1} c(e) \leq (k-k_1)\cdot 2\sum_{e\in T_2} c(e)$$

Az első egyenlőtlenségnél c_{min}-nel, azaz a legrövidebb $k-k_1$ hosszú út költségével becsülünk alul minden lehetséges utat, amikből persze k_2-k_1 darab van. A másik egyenlőtlenség, azért teljesül, mert hozzávesszük még azon élek költségét is, amelyek T_1-beliek is. A 2-es szorzó pedig az élek duplázása miatt indokolt. Ebből kapunk egy felső becslést:

$$c_{min} \leq 2\frac{k-k_1}{k_2-k_1}\sum_{e\in T_2} c(e)$$

Annak az élnek a költsége, amivel hozzákapcsoljuk az utunkat a gyökérhez legfeljebb c_{opt} a fejezet elején található második feltételezésünk szerint. Egy észrevételre van már csak szükségünk és készen állunk az approximáció bizonyítására.

$$\frac{k-k_1}{k_2-k_1} = \frac{k-(n-|A_1|)}{(n-|A_2|)-(n-|A_1|)} = \frac{|A_1|-(n-k)}{|A_1|-|A_2|} = \alpha_2$$

Most az eddigieket felhasználva:

$$\begin{aligned}\sum_{e\in T_1} c(e) + 2\alpha_2\sum_{e\in T_2} + c_{opt} &\leq 2\left(\sum_{e\in T_1} c(e) + 2\alpha_2\sum_{e\in T_2} c(e)\right) + c_{opt}\\ &\leq 4c_{opt} + c_{opt} = 5c_{opt}\end{aligned}$$

Itt kihasználtuk, hogy $\alpha_2 < \frac{1}{2}$ miatt $\alpha_1 \geq \frac{1}{2}$ illetve használtuk a 5.2.3 lemmát a második egyenlőtlenségnél.

Összefoglalva Garg algoritmusa legrosszabb esetben 5-approximációs, de ha szerencsések vagyunk elérhetünk 4 esetleg 2-approximációt is.

Még foglalkoznunk kell a k-TSP-vel is, amit az eddigiek alapján próbálunk megoldani. Garg algoritmusa ugyanúgy alkalmazható csak a szubrutin hívásainál a PTSP-nél látottak szerint járunk el, azaz minden v csúcsra legyen $\pi(v) = \frac{\lambda}{2}$, majd ezzel a büntetésfüggvénnyel hívjuk meg a GW-algoritmust. A kapott Steiner-fa éleit megduplázzuk, és az így kapott Euler-sétát pedig lerövidítjük. A közelítési számok ugyanazok maradnak.

6. A Quota-TSP és a PCTSP probléma

Nagyon sok esetben ezt a két feladatot ugyanazon problémaként tekintik, hiszen a lényeg mindkét esetben az, hogy egy bizonyos mennyiségű kvótát kell összegyűjteni a túra során. A PCTSP esetében most még azt is hozzátesszük, hogy büntetést is kell fizetnünk, ami egy kicsit megnehezíti a dolgunkat, így ezzel a problémával csak keveset foglalkozunk. A Quota-TSP során, viszont megpróbálunk egy jó approximációs algoritmust adni Garg-algoritmusa segítségével felhasználva a Quota-MST problémát. Végül ismertetünk két mohó algoritmust is és megvizsgáljuk a hatékonyságukat.

6.1. A Quota-TSP megoldása approximációs eljárással

Ez a rész nagy mértékben felhasználja az előző fejezetben elhangzottakat, így megírásához a [7] cikk is fontos szerepet töltött be. Emellett a [10] cikket is használtam a gondolatmenetem ellenőrzése céljából. Először is definiáljuk a problémákat.

6.1.1. Definíció: (Quota-MST)**.** *Adott egy $G = (V, E)$ irányítatlan gráf az élein egy $c : E \mapsto \mathbb{Q}^+$ költségfüggvénnyel, ami teljesíti a háromszög-egyenlőtlenséget, a csúcsain egy $w : V \mapsto \mathbb{Z}^+$ súlyfüggvénnyel, egy kiválasztott r gyökércsúccsal és adott egy Q egész szám is. A feladatunk olyan r-t tartalmazó F fát találni, amire $\sum_{v \in F} w(v) \geq Q$ és ezen belül minimális költségű c-re nézve.*

6.1.2. Definíció: (Quota-TSP)**.** *Adott egy $G = (V, E)$ irányítatlan gráf az élein egy $c : E \mapsto \mathbb{Q}^+$ költségfüggvénnyel, ami teljesíti a háromszög-egyenlőtlenséget, a csúcsain egy $w : V \mapsto \mathbb{Z}^+$ súlyfüggvénnyel, egy kiválasztott r gyökércsúccsal és adott egy Q egész szám is. A feladatunk olyan r-t tartalmazó T túrát találni, amire $\sum_{v \in T} w(v) \geq Q$ és ezen belül minimális költségű c-re nézve.*

Írjuk fel a Quota-MST-t IP-programként. Hasonlóan leírható, mint a k-MST esetében:

$$\sum_{e \in \delta(U)} x_e + \sum_{U' : U \subseteq U'} x_{U'} \geq 1 \quad \forall U \in \mathcal{F} \tag{6.1}$$

$$\sum_{U \in \mathcal{F}} x_U \sum_{v \in U} w(v) \leq \sum_{v \in V} w(v) - Q \tag{6.2}$$

$$\min \sum_{e \in E} c(e) x_e \tag{6.3}$$

$$x_e, x_U \in \{0,1\} \quad \forall e \in E, \quad \forall U \in \mathcal{F}$$

Egyetlen változás történt mégpedig a (6.2) feltételben, ami garantálja, hogy a bejárt csúcsok súlyainak összege legalább Q. Hasonlóan relaxáljuk a (6.2)-beli feltételt valamilyen $\lambda \geq 0$-val. Az így keresendő Lagrange-primál optimum a következőképpen néz ki:

$$\min \sum_{e \in E} c(e) x_e + \lambda \left(\sum_{U \in \mathcal{F}} x_U \sum_{v \in U} w(v) - \left(\sum_{v \in V} w(v) - Q \right) \right) \tag{6.4}$$

Ebből visszatudjuk vezetni a feladatot a PCSTP feladatra mégpedig úgy, hogy a $\pi(v)$ büntetésfüggvényt definiáljuk úgy, hogy $\pi(v) := \lambda \cdot w(v)$ minden v csúcsra.

Ez alapján felírva a Lagrange-duál optimumot nagyon hasonlóhoz jutunk, mint a k-MST esetében:

$$\max \sum_{U \in \mathcal{F}} y_U - \lambda \left(\sum_{v \in V} w(v) - Q \right)$$

$$\lambda \geq 0, y_U \geq 0$$

Most hasonlóan alkalmazzuk a 5.1.4 következményt és annak mindkét oldalából kivonunk $2\lambda \left(\sum_{v \in V} w(v) - Q \right)$-t.

$$\sum_{e \in T} c(e) + 2\lambda \left(Q^- - \left(\sum_{v \in V} w(v) - Q \right) \right) \leq 2 \left(\sum_{U \in \mathcal{F}} y_U - \lambda \left(\sum_{v \in V} w(v) - Q \right) \right)$$
$$\leq 2 c_{opt}$$

Itt Q^- a kihagyott csúcsok súlyainak összege. Ha a GW algoritmust futatjuk egy rögzített $\lambda \in \mathbb{R}$ esetén és azt kapjuk, hogy $Q^- = \sum_{v \in V} w(v) - Q$, akkor szerencsések vagyunk és ugyanúgy 2-approximációt érünk el. Egyébként pedig Garg-algoritmusához hasonlóan keressük a megfelelő lambdát. A két feltevésünk, amit a k-MST esetében tettünk itt is fennáll. A szubrutin hívások végén megkapjuk a két lambdához tartozó PCSTP megoldást, így egy (T_1, Q_1^-) és (T_2, Q_2^-) párost, ahol Q_i^- az i-edik fa által kihagyott csúcsok összsúlya. Legyen továbbá Q_i az i-edik fa által összegyűjtött kvóta. A leállási

feltételből tudjuk, hogy ekkor $Q_1 < Q < Q_2$. Ezután vesszük a T_i-k konvex kombinációját. A konvex kombináció együtthatóit hasonlóan számoljuk ki. Ezek a következőek lesznek:

$$\alpha_1 = \frac{(W - Q) - Q_2^-}{Q_1^- - Q_2^-} \quad \alpha_2 = \frac{Q_1^- - (W - Q)}{Q_1^- - Q_2^-}$$

Itt $W = \sum_{v \in V} w(v)$, illetve Q_i^- az i-edik fa által kihagyott csúcsok súlyainak összege. Ezután a fenti 5.2.3 lemma ilyen formában is belátható. Ha $\alpha_2 \geq \frac{1}{2}$, akkor hasonlóan látható a 4-approximáció. Ellenkező esetben, viszont kicsit változtatni kell a módszerünkön. Itt is azt fogjuk csinálni, hogy T_1-et kipótoljuk csúcsokkal T_2-ből úgy, hogy a megfelelő kvótát elérjük. Ehhez vesszük azon csúcsokat, amelyek T_2-ben szerepelnek, de T_1-ben nem. Ezek összsúlya legyen $Q_l \geq Q_2 - Q_1$. Megduplázzuk T_2 éleit, majd csinálunk egy Q_l összsúlyú T_l túrát a fent említett módon. Ebben kell majd megkeresnünk egy minimális költségű, $Q - Q_1$ összúlyú utat, amelyet majd hozzákapcsolhatunk T_1-hez. A k-MST során látott algoritmus egy az egyben nem használható ebben az esetben, egy kicsit trükköznünk kell. Csináljuk meg T_l-ből T_l'-t a következő módon. Minden T_l-beli v csúcsnak feleltessünk meg $w(v)$ darab, 1 súlyú csúcsot (ez megtehető, mert $w(v)$ pozitív egész), amelyeket 0 költségű élekkel kapcsoljunk egymáshoz úgy, hogy egy utat kapjunk. Az út két végpontjához pedig kapcsoljuk azon éleket, amelyek a túrában az adott v csúcshoz kapcsolódtak és legyen ugyanaz az élköltségük mint T_l-ben. Ekkor T_l' egy olyan Q_l csúcsú gráf, aminek az összsúlya és az összköltsége megegyezik T_l-ével. Megfigyelhető, hogy egy T_l'-beli útnak megfeleltethető egy ugyanolyan költségű T_l-beli út, amely legalább annyi kvótát gyűjt össze. A feladatunk tehát T_l'-ben egy minimális költségű $Q - Q_1$ csúcsból álló út keresése. Ennek az útnak a költségére már alkalmazható a fent látott becslés, így egy ilyen útnak a költségére, c_{min}-re fenáll a következő:

$$c_{min} \leq 2\frac{Q - Q_1}{Q_2 - Q_1} \sum_{e \in T_2} c(e)$$

Innentől kezdve már minden ugyanúgy működik. Az így kapott utat hozzákapcsoljuk egy éllel T_1-hez. Észrevehetjük, hogy $\frac{Q-Q_1}{Q_2-Q_1} = \alpha_2$, majd a feltevésünk és a fő lemma használatával belátható, hogy az így kapott fa költsége nem lehet rosszabb mint az optimum 5-szöröse, így ez az algoritmus is 5-közelítő. Az előző fejezetben látott gondolatmenettel tudunk 5-approximációs algoritmust adni a Quota-TSP megoldására is.

6.2. Mohó algoritmusok a Quota-TSP megoldására

A következőekben két mohó algoritmust ismertetünk a Quota-TSP megoldására, amelyek egyszerűnek tűnhetnek, de sokszor elég jó eredményt adnak. A végén majd a két algoritmust hatékonysági szempontból is összehasonlítjuk. A Quota-TSP-nek azt a változatát tekintjük, amelyben előre adott egy r gyökércsúcs is. Ez feltehető, mert különben futtatjuk az algoritmust az összes lehetséges gyökércsúcs választással, és az így kapott eredményekből a legkisebb költségű megoldást választjuk. Ebben a részben az algoritmusokat a [8] cikk alapján ismertetjük. A kiszámított eredményeket, viszont egy általam készített program alapján fogom közölni.

Algoritmus 1

Ez az algoritmus lényegében azt csinálja, hogy kiindul a gyökérből és ha egy adott csúcsban van, akkor kiválasztja a legnagyobb súlyú még meg nem látogatott szomszédját és arra lép tovább. Ez addig megy, amíg a súlyok összege már nem kisebb, mint a megkövetelt kvóta. A végen kapunk egy gyökérből induló utat, majd ehhez hozzávesszük azt az élt az outputunkhoz, amelyik a gyökeret köti össze azzal a csúccsal, ahol megálltunk, így kapva egy túrát. Most lássuk ezt precízebben.

Input: egy $G(V, E)$ irányítatlan teljes gráf, egy $r \in V$ gyökércsúcs és egy Q egész szám.

Output: egy T túra, aminek költsége C és a súlya P

Jelölések:

- T az épülő út csúcssorrenddel megadva, ami kezdetben üres
- s az a csúcs, ahol az algoritmus éppen jár
- u az a csúcs, ahova az algoritmus lépni fog
- U azon halmaz, ami a még meg nem látogatott csúcsokat tartalmazza
- C az eddig megtalált túra költsége
- P az eddig megtalált túra súlya

Nézzük most az algoritmus pszeudokódját:

1. $C, P := 0, \quad T := \emptyset, \quad s := r, \quad U := V \setminus \{r\}$ (inicializálás)

2. **while** $(P < Q)$

3. $u := \arg\max_{v \in U} w(v)$

4. $U := U \setminus \{u\}, \quad T = T \cup \{u\}$

5. $C = C + c(su), \quad P = P + w(u)$

6. $s = u$

7. **end**

8. $C = C + c(sr)$

9. **RETURN**(T, C, P)

Most lássunk egy olyan algoritmust, ami az elözőhöz nagyon hasonló, de jóval hatékonyabb, amiről később meg is győződhetünk.

Algoritmus 2

Sokban nem fog különbözni az előzőtől, csak a továbblépést kicsit ügyesebben választjuk meg. Felhasználjuk az élek költségét is a maximalizáláshoz. A célunk az, hogy minél nagyobb súlyú csúcsot vegyünk be az épülő utunkba, de az odavezető él költsége se legyen olyan nagy. Ezt a következőképpen tesszük meg: azt a csúcsot választjuk az út folytatásaként, amely maximalizálja a súlyának és az odavezető él költségének hányadosát. Később megfigyeljük, hogy ilyen kis változtatás milyen mértékű eltérést okozhat a két algoritmus futásánál. Ekkor pont egy számunkra ideális csúcshoz jutunk. Ha esetleg 0 költségű éllel is találkozunk az algoritmus során, akkor természetesen használjuk azt. Persze a gyakorlatban ez nem sokszor fordul elő. A pszeudokódunk egy sorban módosul mégpedig a 3. sor esetében:

$$u := \arg\max_{v \in U} \frac{w(v)}{c(sv)}$$

A két algoritmus MATLAB kódját a függelékben megtalálhatjuk, ha kíváncsiak vagyunk a számítógépes implementálásukra is.

A fejezet zárásaként összehasoníltjuk a két algorimtust két szempontból is. Adott csúcsszám és adott kvóta esetén megvizsgáljuk, hogy az algoritmusaink által adott túrák költsége és súlya hogy viszonyul egymáshoz. Az élköltségekhez és a kvótákhoz véletlenül generálunk 1 és 10 közti egész

számokat. A következő táblázat azt mutatja, hogy adott csúcsszám és kvóta mellett milyen költségű megoldást ad a két algoritmus. Értelemszerűen az adatpárok első tagjai az egyszerű mohó algoritmusra vonatkoznak, a második tagjai pedig a javítottra.

$\|V\| \setminus Q$	120	130	140	150	160
40	[86,37]	[100,45]	[121,49]	[112,28]	[124,52]
50	[81,18]	[96,43]	[86,31]	[114,55]	[131,60]
60	[83,36]	[112,58]	[86,55]	[88,36]	[101,33]
70	[78,45]	[83,22]	[96,26]	[99,23]	[101,28]
80	[79,32]	[114,40]	[113,23]	[97,37]	[113,24]

1. táblázat. A megoldások költségeinek változása a két algoritmus során

Megfigyelhető, hogy mindegyik esetben a javított algoritmusunk ad jobb megoldást, sőt a csúcsszám és a kvóta növekedésével a költségek közti különbség is egyre nagyobb lesz átlagosan, így nagy csúcsszám vagy kvóta esetén sokkal jobban megéri a javított verziót használni. Most nézzük meg mi a helyzet a kvótákkal. Ennek eredményeit a következő táblázat mutatja.

$\|V\| \setminus Q$	120	130	140	150	160
40	[120,122]	[133,131]	[142,140]	[153,155]	[161,162]
50	[126,122]	[133,137]	[144,142]	[151,152]	[165,168]
60	[124,121]	[130,131]	[141,142]	[153,153]	[165,162]
70	[126,124]	[136,132]	[145,145]	[150,152]	[168,160]
80	[126,121]	[136,132]	[142,144]	[152,157]	[167,165]

2. táblázat. A megoldások súlyainak változása a két algoritmus során

Jól látható, hogy a két algoritmus nagyon hasonló eredményeket ad, amelyek a szükséges kvótához viszonylag közel vannak.

6.3. A PCTSP probléma

A PCTSP probléma lényegében egy közös változata a PTSP és a Quota-TSP problémáknak, éppen ezért nagyon nehéz rá jó közelítő algoritmust adni. Az ötletet a probléma megoldására a [5] cikk alapján közöljük. Lássuk a probléma leírását:

6.3.1. Definíció:. *Adott egy $G = (V, E)$ irányítatlan gráf az éleken egy $c : E \mapsto \mathbb{Q}^+$ költségfüggvénnyel, ami teljesíti a háromszög-egyenlőtlenséget, a csúcsain egy $\pi : V \mapsto \mathbb{Q}^+$ ún. büntetésfüggvénnyel" illetve egy $w : V \mapsto \mathbb{Z}^+$ súlyfüggvénnyel. A csúcsok között van egy kitüntetett $r \in V$, amelyet gyökérnek nevezünk. Adott még egy Q egész szám is. A feladatunk találni egy olyan legalább Q súlyú $T = (V_T, E_T)$ túrát, amely tartalmazza r-et és minimalizálja a következő kifejezést:*

$$c(T) = \sum_{e \in E_T} c(e) + \sum_{v \in V \setminus V_T} \pi(v)$$

A probléma megoldására csak egy ötletet mondunk, ami a [5] cikkben is olvasható. Az alapötlet lényegében annyi, hogy a gráfunkon lefutattunk egy α-approximációs algoritmust a PTSP megoldására úgy, hogy a súlyokkal nem foglalkozunk, majd lefutattunk egy β-approximációs algoritmust a Quota-TSP megoldására miközben eltekintünk a büntetésektől. Az így kapott két megoldásból létrehozható egy PCTSP megoldás, amely $\alpha + \beta$ approximációs. A szakdolgozatban tárgyalt algoritmusok alapján kijelenthető, hogy létezik 7-approximációs algoritmus a probléma megoldására.

7. Függelék

A tesztelő függvényünk illetve a mohó algoritmusok kódja:

```
function T = testtable()
T = {5,5};
for i=1:5
    for j=1:5
        T{i,j} = greedy(30 + 10*i,1,110 + 10*j);
    end
end
end
```

```
function W = greedy(n,r,Q) %Adott a csúcszám, a gyökér és a kvóta
A = ones(n)-diag(ones(n,1));
G = graph(A~=0); %Megcsináljuk a teljes gráfot az adjancenciamátrixából
for i=1:n
    for j=i:n
        if i ~= j
            A(i,j) = randi(10,1,1); %Véletlenül választunk élköltségeket
            A(j,i) = A(i,j); %Szimmetrikussá tesszük A-t
        end
    end
end
w = randi(10,n,1); %Véletlenül választunk súlyokat
S=-w(r);
for i=1:n
    S = S + w(i);
end
if S < Q
    W = 'No solution!'; %Leellenőrizzük, hogy megoldható-e a feladat
else
P = 0;
C = 0;
s = r; %Inicializálunk
p = zeros(1,numnodes(G)); % p lesz a látottsági számokat tartalmazó tömb
p(r) = 1; %Kezdetben a gyökér már látott csúcs
T = zeros(1,numnodes(G)); %Ez a tömb adja meg az utunkat csúcssorenddel
```

```
j=1;
T(j) = s; %A kezdő csúcs a gyökér
while P<Q
j=j+1;
N = neighbors(G,s); %Kigyűjtjük s szomszédjait
max = 0;
argmax = 0;
for i=1:length(N)
    if p(N(i)) == 0
        if w(N(i)) > max %Algoritmus2 során: w(N(i))/A(s,N(i)) > max
            max = w(N(i));
            argmax = N(i);
        end
    end
end %Kiválasztjuk azt a csúcsot, amelyikbe az algoritmus során lépnünk kell
p(argmax) = 1;
P = P + max;
C = C  + A(s,argmax);
s = argmax;
T(j) = s; %Továbblépünk a megfelelő csúcsra és belevesszük az útba
end
C = C + A(r,s); %Felvesszük a megfelelő élet, hogy egy túrát kapjunk
W = [T,C,P]; % Az output
end
end
```

Hivatkozások

[1] Király Zoltán. Algoritmuselmélet

[2] Király Tamás, Kis Tamás és Szegő László. Online jegyzet az Egészértékű Programozás I. és II. tárgyhoz

[3] Operációkutatás II. hallgatói jegyzet

[4] Arya, Sunil, and Hariharan Ramesh. „A 2.5-factor approximation algorithm for the k-MST problem." Information Processing Letters 65.3 (1998): 117-118.

[5] Ausiello, Giorgio, et al. „Prize collecting traveling salesman and related problems." Handbook of Approximation Algorithms and Metaheuristics. Chapman and Hall/CRC, 2018. 611-628.

[6] Goemans, Michel X., and David P. Williamson. „A general approximation technique for constrained forest problems." SIAM Journal on Computing 24.2 (1995): 296-317.

[7] Chudak, Fabián A., Tim Roughgarden, and David P. Williamson. „Approximate k-MSTs and k-Steiner trees via the primal-dual method and Lagrangean relaxation." International Conference on Integer Programming and Combinatorial Optimization. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001.

[8] Ruiz, Justin, et al. „Prize-collecting traveling salesman problem: a reinforcement learning approach." ICC 2023-IEEE International Conference on Communications. IEEE, 2023.

[9] Johnson, David S., Maria Minkoff, and Steven Phillips. „The prize collecting steiner tree problem: theory and practice." SODA. Vol. 1. No. 0.6. 2000.

[10] Ausiello, Giorgio, Stefano Leonardi, and Alberto Marchetti-Spaccamela. „On salesmen, repairmen, spiders, and other traveling agents." Italian Conference on Algorithms and Complexity. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000.

NYILATKOZAT

Név: Kiss Bendegúz

ELTE Természettudományi Kar, szak: Matematika BSc

NEPTUN azonosító: PD789C

Szakdolgozat címe:
Kihagyásos túrák és fák

A **szakdolgozat** 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, 2024

a hallgató aláírása