Eötvös Loránd Tudományegyetem
Matematika Intézet

Rónai Sára

A Steiner-fa probléma

BSc szakdolgozat

Témavezető: Szabó Csaba
egyetemi tanár

ELTE Algebra és Számelmélet Tanszék

Budapest 2015.

Tartalomjegyzék

Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani témavezetőmnek, Szabó Csabának, hogy felkeltette érdeklődésemet a téma iránt, és hogy mindig bizalommal fordulhattam hozzá. Tanácsaival, észrevételeivel nagy mértékben hozzájárult a dolgozat elkészítéséhez.

Köszönettel tartozom ezenkívül szüleimnek, testvéremnek, páromnak, barátaimnak, hogy az elmúlt három év alatt mindvégig mellettem álltak, biztattak, támogattak. Nélkülük ez a szakdolgozat nem jöhetett volna létre. Külön szeretném megköszönni testvéremnek, hogy mindig volt kire felnézni, volt honnan erőt meríteni. A matematika iránti szeretete, lelkesedése ösztönzőleg hatott rám.

Bevezetés

A Steiner-fa problémával elsők között a 19. században elő svájci matematikus, Jakob Steiner kezdett el foglalkozni. A főleg geometria iránt érdeklődő tudós azon a feladaton dolgozott, hogy hogyan lehetne három falut a lehető legrövidebb úthálózattal összekötni. A probléma megoldása után az általános esetet is megfogalmazta, ezzel megteremtve a Steiner-fa probléma alapjait. Körülbelül két évszázaddal korábban Fermat, Toricelli és Cavalieri is foglalkoztak evvel a kérdéssel, de Jakob Steiner tőlük függetlenül oldotta meg a feladatot.

Szakdolgozatom három fejezete a Steiner-fa probléma három fő változatát mutatja be. Az elsőben láthatjuk a legáltalánosabb megközelítést, amikor a feladatot gráfokon vizsgáljuk. A második és harmadik fejezet speciálisabb esetekre tér ki. A másodikban az euklideszi síkon értelmezzük a feladatot, a szokásos, geometriából ismeretes távolságfogalmat használva. A harmadik, és egyben utolsó fejezetben olyan fát kell majd megadnunk, melynek élei kizárólag vízszintes és függőleges szakaszokból állhatnak.

Az egyes fejezetekben olyan cikkeket dolgoztunk fel, melyek az adott témában alapvetőnek számítanak. A legfrissebb eredmények is ezekre vezethetőek vissza, illetve számtalan olyan cikket találtunk, melyek ezekre hivatkoznak. Olyan, mondhatjuk úgy, hogy klasszikus algoritmusok kaptak helyet a dolgozatban, melyek nem vesznek el a részletekben, ezzel egy általános képet adnak a témakörről. Továbbá a válogatás során arra is figyeltünk, hogy olyan eredményeket vegyünk sorra, melyek különböző ötletekre épülnek. Remélhetőleg sikerült evvel színesebbé és érdekesebbé tenni a szakdolgozatot.

1. fejezet

A Steiner-fa probléma gráfokon

Ebben a fejezetben a Steiner-fa problémát gráfokon szeretném vizsgálni, így mindenekelőtt tekintsünk át néhány gráfelméleti alapfogalmat, illetve jelölést. A gráfot jelölje $G = (V, E)$, ahol V egy nemüres halmaz, a csúcsoknak a halmaza, E pedig a gráf éleinek a halmaza. A csúcsokat jelöljük $v_1, v_2, ...v_n$-nel, az éleket pedig $\{v_i, v_j\}$ $(v_i, v_j \in V)$ alakú párokkal. A gráf minden éléhez rendeljünk hozzá egy pozitív számot, amit az él súlyának, költségének vagy hosszának szokás nevezni. A hozzárendelést megadó függvényt súlyfüggvénynek hívjuk, és a továbbiakban d-vel jelöljük. Így tehát definiáltunk egy élsúlyozott gráfot, amin további fogalmakat tudunk értelmezni.

A gráf két élét szomszédosnak nevezzük, ha van egy közös csúcsuk. Hasonlóan két csúcs szomszédos, ha éllel vannak összeköve. Szomszédos csúcsok és élek váltakozó sorozatát útnak hívjuk. Ha az útban minden csúcs legfeljebb egyszer szerepel, akkor egyszerű útról beszélünk, és ha az első és az utolsó csúcs is megegyezik, akkor körről. Ha az út az $u_1, u_2 \dots u_p$ csúcsokból - és természetesen $\{u_1, u_2\}, \{u_2, u_3\} \dots \{u_{p-1}, u_p\}$ élekből - áll, akkor az út hosszát a

$$\sum_{k=1}^{p-1} d\left(\{u_k, u_{k+1}\}\right)$$

összeggel definiáljuk.

Egy gráfot összefüggőnek nevezünk, ha bármely két különböző csúcsa között vezet út, és körmentesnek, ha nem tartalmaz kört. Az összefüggő körmentes gráfokat fának nevezzük. A $G' = (V', E')$ gráf a $G = (V, E)$ gráf részgráfja, ha $V' \subseteq V$, $E' \subseteq E$, és egy pont és egy él pontosan akkor illeszkedik egymásra G'-ben, ha G-ben is illeszkedők, azaz a pont az él egyik végpontja. Feszítő részgráfról beszélünk, ha $V' = V$ is igaz, azaz a részgráf G összes pontját tartalmazza. Az F gráf a G gráf feszítőfája, ha F fa és feszítő részgráfja G-nek. A minimális feszítőfa az a feszítőfa, amiben az élek összsúlya a lehető legkisebb.

A minimális Steiner-fa keresése egy élsúlyozott gráfból $(G = (V, E, d))$ és a csúcsoknak egy

tetszőleges nemüres részhalmazából indul ki. Ez a részhalmaz - jelöljük a továbbiakban S-sel - azokat a csúcsokat tartalmazza, amiket az új gráfunkba (G') mindenképpen be kell vennünk. Szokás ezeket a pontokat Steiner pontoknak nevezni. Ezen kívül további V-beli csúcsokat is bevehetünk a gráfba, de ez már nem kötelező. Így kapunk egy csúcshalmazt, ez legyen V'. Ezek után olyan éleket adunk az új gráfhoz, melyek a G gráfnak is élei voltak, és mindkét végpontja benne van V'-ben. Annyi élt kell hozzáadnunk legalább, hogy G' összefüggő legyen. A cél az, hogy minél kisebb összsúlyú legyen ez a G' gráf, így nyilván G'-t érdemes körmentesnek, azaz fának választani. Az így kapott G' gráfot Steiner-fának nevezzük. Formálisan tehát $G' = (V', E', d)$ olyan fa, hogy $S \subseteq V' \subseteq V$, $E' \subseteq E$, és egy pont és egy él pontosan akkor illeszkedik egymásra G'-ben, ha G-ben is illeszkedők, azaz a pont az él egyik végpontja. Az élek hosszát nem változtatjuk meg, így G'-ben is d lesz a súlyfüggvény. A konstrukcióból látszik, hogy $S = V$ esetén a feladat megegyezik a minimális feszítőfa keresésével. Ha azonban S valódi részhalmaza V-nek, akkor rengeteg G' Steiner-fát adhatunk meg. Ezek között a minimális, azaz a legkisebb összsúlyú megtalálása igen bonyolult feladattá válik.

A problémát két irányból is megközelíthetjük. Egyrészt kereshetjük a minimális súlyú Steiner-fát, másrészt tetszőleges adott k szám esetén megkérdezhetjük, hogy létezik-e olyan Steiner-fa, melynek súlya legfeljebb k. Az utóbbi eldöntési feladat Richard Karp híres 21 NP-teljes feladata közé tartozik. 1972-ben megjelent cikkében Cook tételét felhasználva bizonyította, hogy a probléma az NP-teljes problémák családjába tartozik. Ebből adódóan a minimális Steiner-fa keresése, mint optimalizálási feladat NP-nehéz feladat. A tudomány mai állása szerint nincs remény arra, hogy az általános esetben polinom időben futó algoritmussal meg tudjuk keresni a minimális Steiner-fát, vagy el tudjuk dönteni, hogy tetszőleges adott k számhoz létezik-e olyan Steiner-fa, melynek összsúlya legfeljebb k. Emiatt a legtöbb, amit remélhetünk, hogy viszonylag jól tudjuk közelíteni a minimális Steiner-fa súlyát.

1.1. Kou, Markowsky és Berman algoritmusa

Ebben a fejezetben L. Kou, G. Markowsky és L. Berman algoritmusát mutatom be az 1981-ben megjelent cikkük alapján [1]. A szerzők nevének kezdőbetűivel rövidítve KMB algoritmusként is fogok rá hivatkozni. Az előbb látott minimális Steiner-fa keresésre adnak egy közelítő algoritmust, amely legrosszabb esetben - a korábbi jelöléseket használva - $O(|S||V|^2)$ idő alatt fut. Ez azt jelenti, hogy az algoritmus műveletigénye felülről becsülhető $c \times |S| \times |V|^2$-tel valamilyen c konstans mellett. A legfontosabb szempont pedig természetesen a közelítés mértéke. Erre, ahogy később látni fogjuk, egy $|S|$-től függő képletet adnak.

Az algoritmus a következőképpen működik. Bemenetként kap egy élsúlyozott gráfot és a

Steiner pontoknak a halmazát. Első lépésben elkészít egy teljes gráfot, melynek csúcsai a Steiner pontok lesznek. Mivel a gráf teljes, minden pont össze van kötve az összes többi ponttal. Az éleknek a súlya az eredeti gráfban a két végpontot összekötő legrövidebb út hossza lesz. Tehát az így kapott gráfban egy élt feleltetünk meg egy útnak, mégpedig az él két végpontját összekötő legrövidebb útnak. Megjegyezzük, hogy melyik élhez melyik út tartozott, mivel a későbbiekben erre szükségünk lesz. Utána ebben a gráfban keresünk egy minimális feszítőfát. Ezt például Kruskal mohó algoritmusával megtehetjük. Azaz minden lépésben válasszuk ki, és adjuk a fához a legkisebb súlyú olyan élek egyikét, amely még nem alkot kört az eddig kiválasztottakkal. Ha ilyet már nem találunk, akkor megállunk, ha találunk, akkor megismételjük az eljárást. Az így kapott fát úgy alakítjuk tovább, hogy az éleit lecseréljük a korábban nekik megfeleltetett legrövidebb utakra, majd az így kapott gráfban is megkeresünk egy minimális feszítőfát. Ebben a feszítőfában vizsgáljuk utána a leveleket. Ha olyan levelet találunk, amelyik nem Steiner pont, akkor azt töröljük a hozzá csatlakozó éllel együtt. Ezt egészen addig csináljuk, amíg minden levél Steiner pont nem lesz. Az így kapott fa lesz az algoritmus kimenete. Könnyen látható, hogy a kapott fa a Steiner-fa tulajdonságokat teljesíti, hiszen összefüggő, S minden csúcsát tartalmazza és fa.

A következő szakaszban a szavakkal megfogalmazott algoritmust formalizáljuk. Adott $G = (V, E, d)$ élsúlyozott gráf és $S \subseteq V$ Steiner pontok halmaza mellett definiáljuk a $G_1 = (V_1, E_1, d_1)$ gráfot a következőképpen. Legyen $V_1 = S$, és minden csúcsot kössük össze az összes többi csúccsal, azaz legyen a gráf teljes. Minden $\{v_i, v_j\} \in E_1$-re $d(\{v_i, v_j\})$ legyen a v_i-ből v_j-be érkező legrövidebb út hossza.

A KMB algoritmus:

INPUT: $G = (V, E, d)$ élsúlyozott gráf és a Steiner pontok halmaza: $S \subseteq V$
OUTPUT: T_H, a közelítő Steiner-fa

1. lépés: Készítsük el a fent definiált $G_1 = (V_1, E_1, d_1)$ gráfot.

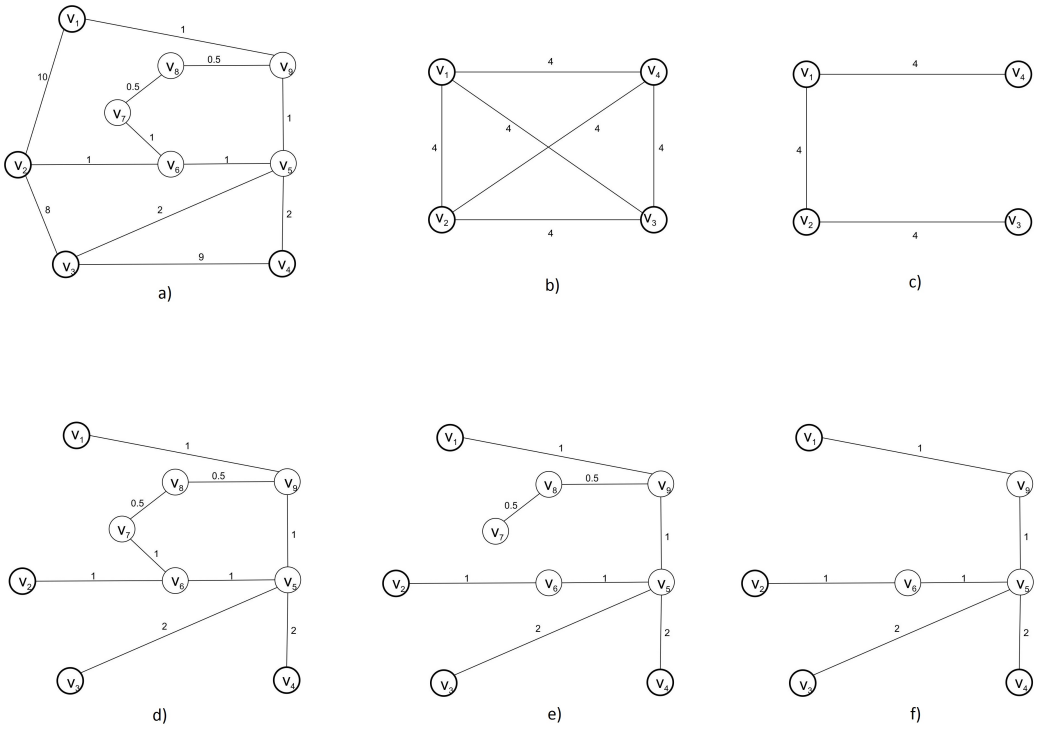
2. lépés: Keressük meg G_1-nek a minimális feszítőfáját, ez legyen T_1. (Ha több minimális feszítőfa van, válasszunk egy tetszőlegeset.)

3. lépés: Készítsük el G-nek azt a G_S részgráfját, melyet úgy kapunk, hogy T_1 éleit lecseréljük a nekik megfeleltetett G-beli legrövidebb utakra.

4. lépés: Keressük meg G_S-nek a minimális feszítőfáját, ez legyen T_S.

5. lépés: T_S-ből addig töröljük az éleket, amíg minden levél Steiner pont nem lesz. Az így kapott fa lesz T_H, a közelítő Steiner-fa.

Az 1.1 ábrasorozat a KMB algoritmus egy lehetséges működését mutatja be. Az a) ábrán látható a bemenetként megadott gráf. Az éleken fel vannak tüntetve a súlyok, a Steiner pon-



1.1. ábra.

tok halmaza pedig álljon a v_1, v_2, v_3 és v_4 csúcsokból. A b)-f) ábrákon az algoritmus lépéseit követhetjük végig.

Ezek után elemezzük az algoritmust. Semmit sem érne egy olyan megoldás, amely kis mértékben közelítené a minimális Steiner-fát, vagy amely nagyon lassan dolgozna. Ez utóbbi alatt azt értjük, hogy a gyakorlatban azok az algoritmusok használhatóak, melyek futásideje az inputuk méretének valamilyen polinom függvénye. Az erre vonatkozó állítás a következőképpen néz ki:

Tétel. *A KMB algoritmus futásideje* $O(|S||V|^2)$.

Bizonyítás. A bizonyításhoz tekintsük az algoritmus lépéseinek futásidejét egyenként. Az első lépésben minden csúcspárra szeretnénk meghatározni a közöttük haladó legrövidebb út hosszát. Ezt megtehetjük úgy, hogy egy csúcsot rögzítünk, és ebből a csúcsból kiindulva futtatjuk a Dijkstra algoritmust. Ekkor az összes olyan csúcspárra megkapjuk a legrövidebb út hosszát, melyben az egyik csúcs a rögzített. Utána rögzítünk egy másik csúcsot, és megismételjük az eljárást. Vegyük észre, hogy ha a gráf minden csúcsa bekerül a rögzített csúcs szerepébe, akkor az összes csúcspár közötti legrövidebb utat megkapjuk. Ez azt jelenti, hogy az algoritmust $|V_1| = |S|$-szer kell futtatnunk. Mivel a Dijkstra algoritmus lépésszáma $O(|V|^2)$, az első lépés futásideje $O(|S||V|^2)$ lesz. A második lépésben egy $|S|$ csúcsú gráfban keresünk a Kruskal algoritmussal

egy minimális feszítőfát. A Kruskal algoritmus műveletigénye ekkor $O(|S|^2 log|S|^2)$, amely a tétel belátásához nem elég. Azonban ha M. L. Fredman és R. E. Tarjan megoldását használjuk a minimális feszítőfa megkeresésére [3] , akkor $O(|S|log|S| + \frac{|S|^2}{2})$ művelet is elegendő lesz. Ez a módszer azonban mélyebb matematikai eszközöket igényel, így a bemutatására nem térünk ki. A harmadik lépés megvalósítható úgy, hogy az eredeti gráf $O(|V|^2)$ élén végigmegyünk, és mindegyikről eldöntjük, hogy szerepel-e valamelyik két pont közötti legrövidebb útban. Tehát a harmadik lépés futásideje $O(|V|^2)$. A negyedik lépés műveletigénye a második lépésnél látott gondolatmenet miatt $O(|V|log|V| + \frac{|V|^2}{2})$, hiszen ennek a gráfnak legfeljebb $|V|$ darab csúcsa van. Az ötödik lépésben pedig $O(|V|)$ művelet kell, hiszen a gráf levelein - melyekből $O(|V|)$ darab van - kell végigmenni és eldönteni, hogy Steiner pont-e vagy sem, és ha nem, törölni. Az előbbieket összegezve azt kapjuk, hogy a KMB algoritmus lépéseinek számát $c \times |S| \times |V|^2$-tel tudjuk felülről becsülni, ahol c alkalmas konstans. Így világos, hogy az algoritmus műveletigénye $O(|S||V|^2)$. □

A továbbiakban térjünk rá arra, hogy az algoritmus mennyire ad jó közelítést. Jelöljük D_H-val a T_H Steiner-fa éleinek összhosszát. Legyen D_{MIN} a minimális Steiner-fa éleinek összhossza. A következő részben egy felső becslést mutatunk a D_H/D_{MIN} hányadosra. Ehhez először a következő tételre lesz szükségünk:

Tétel. *Tekintsünk egy T fát, melynek legalább $m \geq 1$ éle van. Ekkor tudunk mutatni a csúcsoknak egy olyan: $u = u_0, u_1, \ldots u_{2m}$ sorozatát, ahol minden $0 \leq i \leq 2m$-re u_i csúcsa T-nek, és az alábbi két állítás teljesül:*

i) T-nek minden éle pontosan kétszer szerepel. Ezt úgy kell érteni, hogy minden élnek a két végpontja a fenti u sorozatban pontosan kétszer fordul elő egymás mellett.

ii) T-nek minden levele pontosan egyszer szerepel $u = u_0, \ldots u_{2m}$ sorozatban, ha $u_0 = u_{2m}$-t csak egyszer számoljuk. Továbbá ha u_i és u_j két olyan levél a T fában, hogy az u sorozatban közöttük nem szerepel harmadik levél, akkor $u_i, u_{i+1}, \ldots u_j$ egy egyszerű út.

Bizonyítás. Teljes indukcióval bizonyítunk. Legyen $m = 1$, és a fában szereplő egyetlen él két végpontja u_1 és u_2. Ekkor u-nak egy három hosszú sorozatnak kell lennie, legyen ez $u = u_1, u_2, u_1$. A fa egyetlen éle valóban kétszer szerepel a sorozatban, és minden levél egyszer szerepel a fenti megállapodás miatt (hiszen u kezdő és végpontját csak egyszer számoljuk). u_1 és u_2, valamint u_2 és u_1 két-két olyan levél, melyek között a sorozatban nem szerepel harmadik levél. Itt u_1, u_2, valamint u_2, u_1 egy-egy él, ami nyilván egyszerű út is egyben. Ezzel az állítást beláttuk az $m = 1$ esetre. Ezek után tegyük fel, hogy $m = k \geq 1$-re az állítás igaz. Nézzük az $m = k + 1$ esetet. Legyen v_p T-nek egy tetszőleges levele, legyen $\{v_p, v_q\}$ a belőle induló él. Tekintsük a T' fát, melyet úgy kapunk, hogy T-ből elhagyjuk a $\{v_p, v_q\}$ élt. Ekkor T'-nek

eggyel kevesebb, azaz k csúcsa van, T'-re alkalmazhatjuk az indukciós feltételt. Eszerint van egy $u' = u'_0, u'_1 \dots u'_{2k}$ sorozata a csúcsoknak T'-ben, ami teljesíti a fent megfogalmazott állítást. Keressük meg a sorozatban v_q-t, és cseréljük le a v_q, v_p, v_q három hosszú sorozatra. Az így kapott sorozatot jelöljük u-val és vizsgáljuk meg. A sorozat minden eleme csúcsa T-nek, hiszen a T'-ben szereplő csúcsok és az új v_p pont egyaránt csúcsai T-nek. Az újonnan hozzáadott él kétszer szerepel a sorozatban és u'-ben is minden él kétszer szerepelt, így u-ban is minden él pontosan kétszer szerepel. Az új levél, v_p pontosan egyszer szerepel u-ban, v_q nem baj, hogy legalább kétszer szerepel, hiszen már biztosan nem levele T-nek, a többi T'-beli levél előfordulási számát pedig nem növeltük, azaz ők is pontosan egyszer szerepelnek. Az ii) állítás második részének belátásához különböztessünk meg eseteket. Ha két olyan levelet választunk, mely v_p előtt vagy után szerepel a sorozatban, akkor közöttük csak T'-beli csúcsok szerepelnek, tehát az indukciós feltétel miatt rájuk igaz az állítás. Ha a két levelet úgy választjuk, hogy az egyik v_p előtt, a másik v_p után szerepel a sorozatban, akkor közöttük biztosan szerepel egy harmadik levél, mégpedig v_p. Tehát az ilyen párokkal nem kell foglalkozni. A harmadik eset az, amikor az egyik levél v_p, és a másik levél, x legyen olyan, hogy ne szerepeljen közöttük harmadik levél. Legyen x az a levél, amelyik előtte szerepel u-ban és hozzá a legközelebb van. Mivel $u_0 = u_{2m}$, a sorozat tulajdonképpen egy körnek felel meg. Így feltehető, hogy x éppen u_0. Az u sorozatban nem szerepel közöttük harmadik levél, tehát azt kell belátni, hogy $x = u_0, u_1, \dots v_q, v_p$ egyszerű út. T'-nek legalább 2 levele van, u'-ben x után következő első levelet jelölje y. Ha y az u sorozatban v_p után szerepel, akkor az x-et és y-t összekötő út: $x = u_0, u_1, \dots v_q, \dots y$ egyszerű út, azaz minden csúcs egyszer szerepel benne. Így $x = u_0, u_1, \dots v_q, v_p$ is egyszerű út, mivel az előbbinek egy kezdőszelete kiegészítve v_p-vel. A másik eset az, hogy $y = v_q$. Ekkor hasonlóan $x = u_0, u_1, \dots v_q$ egyszerű út, minden csúcs egyszer szerepel benne. Így $x = u_0, u_1, \dots v_q, v_p$ is egyszerű út lesz. Ezzel az állítást $m = k + 1$-re is beláttuk, így kész a bizonyítás. □

Az előbbi tételre támaszkodva $|S|$ függvényében felső korlátot mutatunk a D_H/D_{MIN} hányadosra:

Tétel. *Legyen az algoritmus bemenete a $G = (V, E, d)$ gráf és S a Steiner pontok halmaza. Legyen T_H a közelítő Steiner fa, az éleinek az összhossza D_H. Legyen a minimális Steiner-fa T_{MIN}, az éleinek az összhossza D_{MIN}. Legyen l a levelek száma T_{MIN}-ben. Ekkor $D_H/D_{MIN} \leq 2(1 - \frac{1}{l}) \leq 2(1 - \frac{1}{|S|})$.*

Bizonyítás. Alkalmazzuk az előbbi tételt T_{MIN}-re, csúcsainak a számát jelölje k. Ekkor létezik a csúcsoknak egy olyan $L = u_0, u_1, \dots u_{2k}$ sorozata, melyre igaz, hogy

i) T_{MIN} minden éle pontosan kétszer szerepel L-ben és

ii) T_{MIN} minden levele pontosan egyszer szerepel az L sorozatban. Továbbá ha u_i és u_j két

olyan levél, hogy az L sorozatban nem szerepel közöttük harmadik levél, akkor $u_i, u_{i+1} \dots u_j$ egy egyszerű út.

Ha az utóbbi állítást kicsit más oldalról közelítjük meg, akkor L-et elő tudjuk állítani l darab egyszerű út uniójaként a következő módon. Válasszuk ki a gráf egyik tetszőleges levelét, és vegyük az L-ben utána következő első levelet. E két levél közötti élek alkotják az első egyszerű utat. A második utat a második levél és az L-ben utána következő, harmadik levél közötti élek alkotják. Mivel a fának l levele van, így éppen l darab egyszerű utat kapunk. Töröljük L-ből az így kapott egyszerű utakból a leghosszabbat. Így egy olyan P csúcssorozatot kapunk, melyet $u_0 = u_{2k}$ miatt egy útnak is tekinthetünk. Ha a törölt rész $u_a, \dots u_b$, akkor $P = u_b, \dots u_0 = u_{2k}, \dots u_a$. Erre a P-re igazak a következő állítások:

i) összhossza legfeljebb L összhosszának az $(1 - \frac{1}{l})$-szerese és

ii) T_{MIN} minden éle legalább egyszer szerepel P-ben.

Az első állítás belátásához használjuk a skatulyaelvet. A leghosszabb út hossza minimum $\frac{L}{l}$ lesz. Ennyivel tudjuk tehát legalább P hosszát csökkenteni. Így P hossza kisebb vagy egyenlő, mint L hossza $-\frac{1}{l} \times L$ hossza. Az utóbbi állítás azért igaz, mert egy egyszerű utat töröltünk, melyben minden él legfeljebb egyszer szerepelhet. L-ben minden él pontosan kétszer szerepelt, így P-ben T_{MIN} minden éle legalább egyszer szerepel.

Becsüljük meg P összhosszát felülről. Ehhez arra van még szükségünk, hogy L összhossza $= 2 \times T_{MIN}$ összhossza, ami nyilván teljesül, mert L-ben T_{MIN} minden éle pontosan kétszer szerepel. Így tehát

P összhossza $\leq (1-\frac{1}{l}) \times L$ összhossza $= (1-\frac{1}{l}) \times 2 \times T_{MIN}$ összhossza $= 2 \times (1-\frac{1}{l}) \times D_{MIN}$.

Legyenek a P-ben előforduló Steiner pontok $w_1, w_2, \dots w_k$ ebben a sorrendben. P összhosszának alsó becsléséhez tekintsük az alábbi egyenlőtlenséget:

P összhossza $\geq G_1 \{w_1, w_2\} \dots \{w_{k-1}, w_k\}$ éleket tartalmazó feszítőfájának összhossza $\geq$ $\geq G_1$ minimális feszítőfájának összhossza $\geq D_H$.

Az első egyenlőtlenség azért igaz, mert T_{MIN}-ben minden levél Steiner pont, így a P útnak mindkét végpontja Steiner pont lesz. Emiatt P Steiner pontok közötti utak uniója, mégpedig w_1 és w_2, w_2 és w_3, $\dots w_{k-1}$ és w_k közötti utak uniója. G_1 csúcsai is pontosan a $w_1, \dots w_k$ csúcsok, de az őket összekötő élek hossza a közöttük vezető legrövidebb út hossza. Ezzel az első egyenlőtlenséget beláttuk. A második egyenlőtlenség triviális, a harmadik pedig magából az algoritmusból következik. Hiszen a T_H fát úgy kapjuk, hogy elhagyhatunk éleket G_S-ből, melynek összhossza megegyezik T_1 összhosszával. Így T_H összhossza $= D_H$ biztosan kisebb egyenlő lesz, mint G_1 minimális feszítőfájának az összhosssza.

Az alsó és a felső becslést összerakva azt kapjuk, hogy $D_H \leq 2 \times (1 - \frac{1}{l}) \times D_{MIN}$. Továbbá $l \leq |S|$ miatt $(1 - \frac{1}{l}) \leq (1 - \frac{1}{|S|})$, így $D_H \leq 2 \times (1 - \frac{1}{|S|}) \times D_{MIN}$. Ez a becslés már nem függ T_{MIN} alakjától, csak a Steiner pontok számától. □

Most pedig mutassuk meg, hogy ha ezt az algoritmust tekintjük, akkor a D_H/D_{MIN} felső korlátját nem lehet jobban megadni, azaz tudunk olyan példát adni az algoritmus futására, melyre a hányados éppen a felső korláttal egyenlő. Ehhez természetesen olyan gráfot kell keresni, melyre a T_{MIN}-t is meg tudjuk könnyen határozni.

Tétel. *Minden $l \geq 2$ pozitív számhoz létezik olyan $G = (V, E, d)$ gráf és $S \subseteq V$ halmaz, hogy a minimális Steiner-fának, T_{MIN}-nek l db levele van, összhossza D_{MIN} és az algoritmus olyan T_H közelítő Steiner-fát ad, melyre $D_H/D_{MIN} = 2(1 - \frac{1}{l})$. Azaz a korábban adott felső korlát éles.*

Bizonyítás. Legyen $l \geq 2$ rögzített. Legyen G az a teljes gráf, melynek csúcsai: $V = \{v_1, v_2, \ldots v_{l+1}\}$ és minden $\{v_i, v_j\} \in E$-re:

$$d(\{v_i, v_j\}) = \begin{cases} 2 & \text{ha } v_i \neq v_{l+1} \text{ és } v_j \neq v_{l+1}, \\ 1 & \text{különben ,} \end{cases}$$

Legyen a Steiner pontok halmaza $S = \{v_1, v_2, \ldots v_l\}$. Ahhoz, hogy D_H/D_{MIN} a lehető legnagyobb legyen D_H-t kell a lehető legnagyobbnak megválasztani. A legrosszab esetben a T_H közelítő Steiner-fa $l-1$ darab 2 hosszúságú élből áll, mégpedig például a $\{v_1, v_2\}, \ldots \{v_{l-1}, v_l\}$ élekből. Ez az eset úgy fordulhat elő, hogy az algoritmus az első lépésében minden Steiner pontpár közötti legrövidebb útnak az őket összekötő élt választja. Könnyű látni, hogy a minimális Steiner-fa, T_{MIN} összhossza nem lehet l-nél kevesebb, mivel l levele van, így legalább l éle, és a d súlyfüggvény definíciója miatt így az összhossz legalább l. Továbbá tudunk olyan Steiner-fát mutatni, melynek az összhossza l, így ez lesz a minimális. A következőképpen néz ki: l darab 1 hosszúságú élből áll, név szerint $\{v_1, v_{l+1}\}, \ldots \{v_l, v_{l+1}\}$-ből. Ekkor az is igaz, hogy valóban l darab levele van. Így $D_H/D_{MIN} = 2 \times (l-1)/1 \times l = 2(1 - \frac{1}{l})$. □

1.2. Mehlhorn algoritmusa

A következő fejezetben röviden Kurt Mehlhorn 1988-as algoritmusáról lesz szó [2]. Az algoritmusa lényegében az előbbi szakaszban bemutatott KMB algoritmus továbbfejlesztése. A KMB algoritmus első lépését változtatja meg, a többit megtartja. A futásidőt evvel $O(|E|+|V|log|V|)$-re csökkenti, ahol $|E|$ a gráf éleinek a száma, $|V|$ pedig a csúcsoknak a száma. Az előző szakaszban szereplő jelöléseket a továbbiakban is ugyanúgy fogom használni.

Az algoritmus alapötlete a következő. Vegyük a Steiner pontokat sorra, és minden Steiner ponthoz készítsünk el egy halmazt. Ezek a halmazok azokból a csúcsokból fognak állni, melyek

az adott Steiner ponthoz vannak a legközelebb. Azt, hogy egy csúcs melyik Steiner ponthoz esik a legközelebb, úgy tudjuk megállapítani, hogy megkeressük azt a Steiner pontot, amelyikhez a csúcsból a legrövidebb úton el tudunk jutni. Az út hossza itt is az útban szereplő élek hosszának az összege lesz. Később megmutatjuk, hogy ezt a Steiner pontot hogyan is lehet kiválasztani. Formálisan minden $s \in S$ csúcshoz készítsük el az $N(s)$ halmazt, mely azokból a V-beli csúcsokból áll, melyek az s csúcshoz esnek a legközelebb. Ha egy $v \in V$ csúcsot ez alapján több $N(s)$ halmazhoz is hozzá tudunk rendelni - azaz két Steiner pont is ugyanolyan közel van a v csúcshoz-, akkor rakjuk ezek közül egy tetszőlegesbe. Így V-t előállíthatjuk az $N(s)$ halmazok partíciójaként: $V = \bigcup_{s \in S} N(s)$, ahol $N(s) \cap N(t) = \emptyset$, ha $s \neq t$. Továbbá az is igaz lesz, hogy ha $d_1(x, y)$ az x-ből y-ba vezető legrövidebb út hossza, akkor $v \in N(s)$ esetén $d_1(v, s) \leq d_1(v, t)$ minden $t \in S$-re.

Készítsük el a G_1' gráfot a következő módon. Csúcshalmaza legyen a Steiner pontok halmaza, azaz S. Az E_1' élhalmaz megadásához vegyünk két tetszőleges Steiner pontot. Legyenek ezek s és t, és a hozzájuk rendelt halmazok $N(s)$ és $N(t)$. Ha találunk olyan élt G-ben, melynek egyik végpontja $N(s)$-ben, a másik $N(t)$-ben van, akkor az $\{s, t\}$ élt behúzzuk a G_1' gráfban, ha nincs ilyen, akkor pedig nem. Ezt formulával a következőképpen adhatjuk meg:

$$E_1' := \{\{s, t\} : s, t \in S \text{ és van egy } \{u, v\} \in E \text{ él úgy, hogy } u \in N(s), v \in N(t)\}$$

Így lehet két olyan csúcsa a G_1' gráfnak, melyek között nem vezet él. Az is elképzelhető, hogy egy $\{s, t\}$ élhez több olyan $\{u, v\}$ él is tartozik, melyre teljesül, hogy $u \in N(s)$ és $v \in N(t)$. Ezért a d_1' súlyfüggvényt a következó módon érdemes definiálni:

$$d_1'(s, t) := \min \{d_1(s, u) + d(u, v) + d_1(v, t); \{u, v\} \in E, u \in N(s), v \in N(t)\}$$

Tehát az s-et és t-t összekötő él hossza az s-ből u-ba vezető legrövidebb út hosszának, az $\{u, v\}$ él hosszának és a v-ből a t-be vezető legrövidebb út hosszának az összegének a minimuma lesz. Belátható, hogy az így kapott G_1' gráf minden minimális feszítőfája minimális feszítőfája G_1-nek is. Ennek a bizonyítására most nem térünk ki.

A KMB algoritmus első lépését módosítsuk úgy, hogy $G_1 = (V_1, E_1, d_1)$ gráf helyett a fenti jelölésekkel definiált $G_1' = (S, E_1', d_1')$ gráfot hozzuk létre. Becsüljük meg az így kapott algoritmus futásidejét. Vizsgáljuk meg az algoritmus lépéseinek műveletigényét egyenként. A gondolatmenet egyes részei megegyeznek majd a KMB algoritmus futásidejének bizonyításánál látott elemekkel. Minden $\{u, v\} \in E$ él legfeljebb egy darab E_1'-beli élt definiálhat, így G_1' gráfnak legfeljebb $|E|$ éle lehet. Tehát G_1'-nek a konstrukció miatt $O(|E|)$ éle van. Az algoritmus második lépésében ebben a gráfban keresünk egy minimális feszítőfát. Ezt M. L. Fredman és R. E. Tarjan algoritmusa alapján $O(|S|log|S| + |E|)$ idő alatt végezhetjük el [3]. A harmadik

lépés megvalósítható úgy, hogy az eredeti gráf $O(|V|^2)$ élén végigmegyünk, és mindegyikről eldöntjük, hogy szerepel-e valamelyik két pont közötti legrövidebb útban. Így a futásidő $O(|V|^2)$. A negyedik lépés műveletigénye a második lépéshez hasonlóan $O(|V|log|V| + |E|)$, hiszen a G_S gráfnak $|V|$ csúcsa és $O(|E|)$ éle van. Az ötödik lépésben pedig $O(|V|)$ művelet kell, hiszen a gráf levelein kell végigmenni és eldönteni, hogy Steiner pont-e vagy sem, és ha nem, törölni. A levelek száma pedig éppen $O(|V|)$. Tehát már csak az első lépés futásidejére kell becslést adnunk. Ehhez nézzük meg, hogy a G_1' gráf elkészítéséhez mennyi műveletre van szükség. Először létre kell hoznunk a csúcsoknak a $\{N(s), s \in S\}$ partícióját. Vegyünk a gráfhoz egy s_0 csúcsot, melyet kössünk össze minden $s \in S$ csúccsal. Az új éleken mindenütt 0-nak definiáljuk az él hosszát. Az így kapott gráfon s_0 csúcsból kiindulva futtassunk egy legrövidebb útkereső algoritmust. Ez minden v csúcshoz megkeresi az s_0-ból induló, v-be érkező (egyik) legrövidebb utat. Ez az út pontosan egy Steiner ponton fog átmenni, hiszen ha többet is érintene, akkor az út hosszabb lenne biztosan, mintha v-ből indulva az első Steiner pont elérése után rögtön s_0-ba mennénk. Mivel az s_0-ból induló minden él hossza 0, így meg tudjuk adni minden v csúcshoz azt az $s(v) \in S$ csúcsot, melyre $v \in N(s(v))$. Ezen kívül a legrövidebb út hossza éppen $d_1(v, s(v))$. Az s_0-ból induló legrövidebb utak megkeresése Fredman és Tarjan algoritmusa alapján $O(|V|log|V|+|E|)$ műveletet vesz igénybe. Ezek után menjünk végig az $(u, v) \in E$ éleken és készítsük el az $(s(u), s(v), d_1(s(u), u) + d(u, v) + d_1(v, s(v)))$ hármasokat. Ezeket az első két komponens szerint rendezzük edényrendezéssel, és válasszuk ki minden G_1'-beli él minimális súlyát. Evvel d_1'-t is meghatároztuk, mégpedig $O(|E|)$ idő alatt. Összefoglalva azt kaptuk, hogy az algoritmus minden lépésének műveletigénye felülről becsülhető alkalmas c konstans mellett $c \times (|V|log|V| + |E|)$-vel.

Az eredményeket a következő tétel foglalja össze:

Tétel. *Adott $G = (V, E, d)$ élsúlyozott gráf és $S \subseteq V$ csúcshalmaz esetén Mehlhorn algoritmusa $O(|V|log|V| + |E|)$ idő alatt tud szerkeszteni olyan Steiner-fát, melynek összhossza legfeljebb $2(1 - \frac{1}{l})$-szerese a minimális Steiner-fa összhosszának.*

2. fejezet

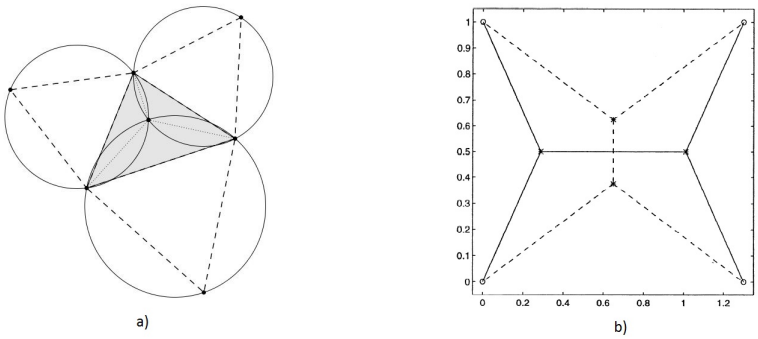
Az euklideszi Steiner-fa probléma

Euklideszi síknak nevezzük a hagyományos, a mindennapi életben használt síkot. Az euklideszi sík pontjainak rendezett számpárok felelnek meg, így a síkot R^2-tel szokták azonosítani. A Steiner-fa probléma esetében adott az euklideszi síkon pontoknak egy halmaza: $T = \{t_1, t_2, \ldots t_n\}$. Ezeket a későbbiekben fix pontoknak fogjuk nevezni. A feladat az, hogy olyan fát keressünk, melynek az összhossza a lehető legrövidebb, és $t_1, t_2, \ldots t_n$ a csúcsai. Itt a hossz alatt az euklideszi távolságot értjük, azaz a fa egy élének a hossza a két végpontját összekötő szakasz hossza. Továbbá megengedett, hogy tetszőleges számú pontot, úgynevezett Steiner pontokat vegyünk fel a síkon, melyek a legkisebb összhosszú fa csúcsai lesznek.

Az euklideszi Steiner-fa probléma a korábban gráfokon értelmezett feladat speciális esetének is tekinthető. Ugyanis a fix pontok és a Steiner pontok alkotják a gráf csúcshalmazát, a pontokat összekötő szakaszok az élhalmazt, egy él hosszán pedig a két végpontját összekötő szakasz hosszát értjük.

Ahhoz, hogy könnyebben el tudjuk képzelni a problémát, vegyünk egy példát a hétköznapi életből. Magyarország meg szeretné építeni az M3-as autópályát Budapest, Miskolc, Debrecen és Nyíregyháza között. A cél természetesen az, hogy minél kevesebb alapanyagot, betont kelljen felhasználni, azaz minél kisebb legyen a beruházás költsége. A kérdés az, hogy merre vezessen az út. Ha ma ránézünk a térképre, látjuk, hogy a feladatot úgy oldották meg, hogy két csomópontot is létrehoztak, ahol az út elágazik. Azt is érezzük, hogy ezek a csomópontok nem kerültek rossz helyre (például Szeged mellé), így rövidebb úthálózatot kapunk, mintha Budapestről Miskolcra, onnan Debrecenbe, majd Nyíregyházára vezetne az út. A mi absztrakt modellünkben a városok lesznek a fix pontok, a csomópontok pedig a Steiner pontok.

Az euklideszi Steiner-fa probléma története egészen Fermatig (1601-1655) nyúlik vissza. Ő vetette fel először a következő problémát: keressük meg a síknak azon pontját, melynek a távolságösszege adott három ponttól minimális. Könnyen látható, hogy ez az euklideszi Steiner-fa probléma sepciális esete három pontra. Toricelli geometriai megoldást adott a feladatra. A há-



2.1. ábra.

rom szakaszra kifelé emeljünk egy-egy szabályos háromszöget, majd szerkesszük meg a köréjük írt köröket. A közös metszéspont lesz a minimalizáló pont. A szerkesztés menetét a 2.1/a) ábra mutatja be. A két matematikus emlékére a problémát Fermat-problémának, a minimalizáló pontot pedig Toricelli pontnak nevezik. Érdemes megjegyezni, hogy amennyiben az adott három pont által meghatározott háromszögnek valamely szöge legalább 120 fokos, akkor a Toricelli pont a háromszögön kívül esik, és már nem minimalizáló pont. Ekkor a tompaszögű csúcs a minimalizáló csúcs. Cavalieri megmutatta, hogy a Toricelli pontot az adott három ponttal összekötő szakaszok 120 fokos szöget zárnak be egymással.

Nézzünk egy másik példát, ahol négy pontot adunk meg a síkon, mégpedig úgy, hogy a négy pont egy 1 és 1,3 oldalú téglalap négy csúcsa. Itt találhatunk olyan Steiner-fát, melynek élei a Steiner pontok körül 120 fokos szöget zárnak be egymással, mégsem ez lesz a minimális Steiner-fa. Ezt a fát a 2.1/b) ábra szaggatott vonallal jelöli, míg a minimális Steiner-fát folytonos vonallal.

2.1. Alapvető észrevételek

A szakdolgozat következő három alfejezete Hwang, Richards és Winter Steiner-fákkal foglalkozó könyvének 1.2 és 1.3 fejezetére [5], valamint D. Dreyer és M. Overton cikkére épül [4].

Az első észrevétel a Steiner pontok fokszámára vonatkozik, vagyis arra, hogy a Steiner pontból hány él indulhat ki. A minimális Steiner-fában minden Steiner pont fokszáma legalább három. Hiszen ha egyfokú Steiner pontunk van, akkor azt törölve a fából kisebb súlyú fát kapunk. Ha pedig egy Steiner pontból két él indul ki, akkor a végpontjaikat összekötő szakasz hossza biztos, hogy kisebb vagy egyenlő, mint a Steiner pontból induló két él hosszának az összege. Ez nem más, mint a háromszög egyenlőtlenség.

A második észrevétel egy szögekre vonatkozó kritérium, miszerint semelyik két él nem zárhat be 120 foknál kisebb szöget. Ugyanis ha a v csúcsnál a bezárt szög kisebb mint 120 fok, akkor

a szög két szögszárán válasszunk tetszőlegesen két pontot, legyenek ezek a és b. Ha az abv háromszöget nézzük csak, akkor a fa összsúlyát csökkenthetjük a v' Toricelli pont beszúrásával. Ebből az is következik, hogy minden Steiner pont legfeljebb harmadfokú. Ugyanis ha negyed vagy magasabb fokú lenne, akkor biztosan keletkezne legalább egy 120 foknál kisebb szög. Az előbbivel összevetve azt kapjuk, hogy a Steiner pontok pontosan harmadfokú pontok, és az élek által bezárt szögnek mindenütt 120 fokosnak kell lennie.

A következő tétel a Steiner pontok számára vonatkozik:

Tétel. *Ha a síkon n darab pontot rögzítünk, akkor a minimális Steiner-fának legfeljebb $2n-2$ csúcsa lehet. Azaz a minimális Steiner-fa megszerkesztéséhez legfeljebb n-2 Steiner pontot kell felvennük a síkon.*

Bizonyítás. Tegyük fel, hogy a minimális Steiner-fának k darab Steiner pontja, így összesen $n+k$ csúcsa van. Ekkor az élek száma nyilván $n+k-1$. Korábban láttuk, hogy minden Steiner pont harmadfokú, és minden rögzített pont legalább elsőfokú. Az éleket meg tudjuk úgy számolni, hogy összeadjuk, hogy az egyes csúcsokból hány él indul ki, majd az eredményt elosztjuk kettővel, hiszen így minden élt kétszer számoltunk. Tehát az élek száma legalább $(3k+n)/2$. Ebből azt kapjuk, hogy

$$n+k-1 \geq \frac{3k+n}{2}$$

ami átrendezve:

$$n-2 \geq k$$

A Steiner pontok száma valóban legfeljebb $n-2$ lehet.

□

A gráfokon értelmezett problémához hasonlóan a minimális Steiner-fa megkeresése az euklideszi síkon NP-nehéz feladat. A fejezetben két olyan algoritmust ismertetünk, mely adott ponthalmaz esetén egy olyan Steiner-fát készít el, melynek összhossza közelíti a minimális összhosszát. Ha megtiltjuk a Steiner pontok felvételét a síkon, akkor a feladat a minimális feszítőfa keresése lesz. A két pont távolságán továbbra is az őket összekötő szakasz hosszát értjük. Vizsgáljuk meg a korábban már említett három pontnak az esetét. Tekintsük azt az elhelyezkedést, amikor olyan szabályos háromszöget alkotnak, melynek minden oldala egységnyi hosszú. A minimális feszítőfa ekkor a háromszög tetszőleges két oldala, azaz a hossza $2 \times 1 = 2$. Ha a minimális Steiner-fát nézzük, amit a Toricelli pont hozzáadásával kapunk, akkor annak az összhossza $3 \times 1/\sqrt{3}$. Ilyen módon az összhossz körülbelül 15 százalékkal csökken, általános esetben pedig a változás még csekélyebb. A közelítő algoritmusok éppen emiatt gyakran a minimális feszítőfából indulnak ki, és ezt próbálják meg különböző módszerekkel javítani.

2.2. Steiner pontot beszúró algoritmus

Az elsőként bemutatott algoritmus az előbbi fejezetben tárgyalt szögekre vonatkozó kritériumon alapszik. Láttuk, hogy a minimális Steiner-fában az éleknek mindenütt legalább 120 fokos szöget kell bezárniuk egymással. Tehát ha találunk két olyan élt, melyek 120 foknál kisebb szöget zárnak be egymással, akkor a Steiner-fa biztosan nem minimális összhosszú. Az algoritmus arra törekszik, hogy ezeket a kritikus élpárokat megtalálja, és kijavítsa a hibát. Ha ez sikerül, akkor a szögekre vonatkozó szükséges feltételt teljesíti a kapott fa. Mivel azonban a feltétel nem elégséges, egyáltalán nem biztos, hogy a minimális Steiner-fát megtaláljuk így. Annyit mondhatunk mindössze, hogy olyan Steiner-fát kaptunk, melynek összhossza jól közelíti a minimális összhosszát.

Az algoritmus a következő módon dolgozik. Bemenetként megkapja a síkon lévő pontoknak a halmazát, ezek lesznek az úgynevezett fix pontok. A pontokat a koordinátáikkal könnyen lehet kezelni, így ilyen módon célszerű megadni őket. Első lépésben keresünk egy minimális feszítőfát. Ezt megtehetjük úgy, hogy tetszőleges két pont között lemérjük a távolságot, majd a távolságokat növekvő sorrendbe rakjuk. Minden lépésben adjuk a fához a legrövidebb élt, ha az még nem alkot kört a korábban kiválasztott élekkel. Ha ilyen módon végigmegyünk az éleknek a hossz szerint rendezett sorozatán, akkor megkapjuk a minimális feszítőfát. Ezután a második lépésben a fix pontokat összekötő éleken megyünk végig tetszés szerinti sorrendben, és elvégezzük az alábbi műveleteket. Az iterációban soron következő fix pontokat összekötő élhez megkeressük azt az élt, amelyik vele a lehető legkisebb szöget zárja be. (Minden élnek két végpontja van, a szögeket a két végpontnál tudjuk mérni. Így minden fix pontokat összekötő él kétszer kerül be a ciklusba, először az egyik végpontjánál, másodszor a másik végpontjánál vizsgáljuk majd a szöget.) Ha a legkisebb bezárt szög legalább 120 fok, akkor nincs további teendő, ha viszont kisebb 120 foknál, akkor egy új Steiner pontot veszünk fel a síkon. Ezt az új pontot egyelőre a két él közös végpontjára helyezzük el, azaz a koordinátái meg fognak egyezni a közös végpont koordinátáival. Ha az egymással 120 foknál kisebb szöget bezáró két él végpontjait nézzük, akkor három pontot kapunk. Ezt a három pontot kössük össze az imént elhelyezett új Steiner ponttal, majd töröljük az eredeti két élt. A három új él közül az egyik nulla hosszúságú lesz, mivel végpontjainak koordinátái azonosak. Amikor az élek által bezárt szögeket vizsgáljuk, ezt az élt nem fogjuk semelyik másik éllel összehasonlítani. Hasonlóan a törölt éleket sem vizsgáljuk már természetesen a későbbiekben, akkor sem, ha azok korábban még csak az „egyik végpontjuk” szerint kerültek be a ciklusba. A harmadik, és egyben utolsó lépésben az új pontrendszeren végzünk optimalizálást. Ez azt jelenti, hogy a Steiner pontokat elmozgatjuk úgy, hogy a Steiner pontok körül a szögekre vonatkozó kritérium mindenütt teljesüljön. Azaz mindenütt megszerkesztjük a Toricelli pontot, hogy megkapjuk a Steiner pont

helyét. A következő szakaszban az algoritmus formális leírása következik.

Az algoritmus:

INPUT: $T = \{t_1, t_2, \ldots t_n\}$ ponthalmaz, a fix pontoknak a halmaza.
OUTPUT: Olyan Steiner-fa, melynek összhossza közelíti a minimális Steiner-fa összhosszát.

1. lépés: Keressünk a T ponthalmazhoz egy minimális feszítőfát.

2. lépés: Minden olyan (t_x, t_y) élre, ahol t_x és t_y is fix pont, végezzük el a következő műveleteket:

a) Keressük meg azt a (t_y, t_z) élt, amely a legkisebb szöget zárja be a (t_x, t_y) éllel. t_z fixpont és Steiner pont is lehet.

b) Ha ez a szög kisebb, mint 120 fok, akkor:

- Vegyünk fel egy új Steiner pontot pontosan ugyanoda, ahol t_y is van. Tehát a koordinátái egyezzenek meg t_y koordinátáival. Hívjuk ezeket a csúcsokat s_n-nek $(n = 1, 2, \ldots)$.

- Töröljük a (t_x, t_y) és a (t_y, t_z) éleket. Innentől kezdve ezeket a 2. pontban lévő ciklusban már nem vesszük figyelembe.

- Húzzuk be a (t_x, s_n), (t_y, s_n) és (t_z, s_n) éleket.

3. lépés: Az $s_1, s_2 \ldots$ Steiner pontokat mozgassuk el úgy, hogy teljesüljön mindenhol a Steiner pontok körül a szögekre vonatkozó kritérium.

Az algoritmus 2/b) lépésben a Steiner pont beszúrása után a (t_y, s_n) él hossza nulla lesz. Amikor a 2/a) lépésben a szögeket vizsgáljuk, az ilyen nulla hosszúságú élekkel nem foglalkozunk, hiszen nem tudunk a velük bezárt szögről beszélni.

Nagyon fontos, hogy a második lépésben, ami tulajdonképpen egy ciklus, az éleket „mindkét irányba" figyelembe kell venni. Ez alatt azt értjük, hogy ha az élt már megvizsgáltuk úgy, hogy az egyik végpontját t_x-nek, a másik végpontját t_y-ak tekintettük, akkor még nem vagyunk „készen" azzal az éllel. Úgy is fel kell dolgoznunk az élt, hogy a korábban t_x-ként vizsgált végpontja most t_y lesz, a másik pedig t_y helyett t_x. Tekintsük például azt az egyszerű esetet, hogy van három pontunk: t_1, t_2 és t_3, és a minimális feszítőfa (t_1, t_2) és (t_1, t_3) élekből áll. Ha a ciklusban mindkét élt $t_1 = t_x$ szereposztásban vizsgálunk meg, akkor világos, hogy nem veszünk fel új Steiner pontot a lépések során. Ha azonban úgy is bekerülnek az élek a ciklusba, hogy $t_2 = t_x$ vagy $t_3 = t_x$, akkor a Steiner pont beszúrásával és a megfelelő helyre mozgatásával egy olyan Steiner-fát kapunk, melynek összhossza kisebb, mint a kiindulási minimális feszítőfa összhossza.

Abban az esetben, ha a minimális feszítőfa nem egyértelmű, a kapott Steiner-fa sem lehet egyértelmű. Hiszen más-más feszítőfa az algoritmus különböző futását eredményezi. Attól is

változhat továbbá a kapott fa alakja, hogy a második lépésben az éleket milyen sorrendben tekintjük.

Nézzük meg az algoritmus egy lehetséges lefutását az alábbi 4 pontra: legyen $t_1 = (0;0), t_2 = (0;1), t_3 = (1{,}3;1), t_4 = (1{,}3;0)$. Ekkor a minimális feszítőfa a (t_1, t_2), a (t_2, t_3), és a (t_3, t_4) élekből áll. A második lépésben vizsgáljuk meg először a (t_1, t_2) élt. Azt tapasztaljuk, hogy a (t_1, t_2) és a (t_2, t_3) élek által bezárt szög 90 fok, így az algoritmus beszúrja az s_1 Steiner pontot úgy, hogy koordinátái megegyezzenek a t_2 csúcs koordinátáival. Ezek után törli a (t_1, t_2) és a (t_2, t_3) éleket, majd felveszi az új Steiner ponthoz csatlakozó $(t_1, s_1), (t_2, s_1)$ és (t_3, s_1) éleket. Hasonlóképpen a (t_4, t_3) él vizsgálatánál is azt kapjuk, hogy új Steiner pontot kell felvennünk, hiszen a (t_3, s_1) éllel bezárt szög szintén 90 fok. Ezek után nem marad fix pontokat összekötő él a gráfban, a ciklus véget ér. Így a következő éllista alakult ki: $\{(t_1, s_1), (t_2, s_1), (t_3, s_2), (t_4, s_2), (s_1, s_2)\}$. Az algoritmus harmadik lépésében az s_1 és az s_2 pontokat kell úgy elhelyezni, hogy a belőlük kiinduló három-három él egymással mindenhol 120 fokos szöget zárjon be. Ha ezzel készen vagyunk, megkaptuk a közelítő Steiner-fát.

Az algoritmus első és második lépésének műveletigénye n pont esetén $O(n^3)$ lesz a legrosszabb esetben, de a gyakorlatban általában ennél sokkal gyorsabb a lefutásuk. A harmadik lépés azonban hatékony megvalósítás esetén is legalább annyi ideig tart, mint az első kettő együttvéve. MATLAB programmal megvalósítva a feladatot 100 pont esetén a futásidő körülbelül 40 másodperc lesz, kisebb méretű problémákra pedig pár másodperc alatt végez a program.

Az algoritmus egyszeri futása nem garantálja azonban, hogy a kapott fában minden szög legalább 120 fokos lesz. Gondoljunk a következő esetre: vegyünk fel a síkon 6 pontot úgy, hogy közülük egy „középen” legyen, a többi tőle egységnyi távolságra úgy, hogy ha összekötjük őket a középső ponttal, akkor mindenhol 72 fokos szöget kapjunk. A minimális feszítőfát a külső pontokat a középsővel összekötő élek alkotják. Az algoritmus két 72 fokos szögnél tud javítani, de amikor a Steiner pontok az utolsó lépésben a helyükre kerülnek, a középső pont körül marad még 120 foknál kisebb szög.

Az algoritmus hatékonyságára a 2.4. fejezetben térünk majd ki, ahol a következő szakaszban szereplő növekvő optimalizálási algoritmus hatékonyságával fogjuk összehasonlítani.

2.3. Növekvő optimalizálási algoritmus

Ebben a fejezetben szereplő algoritmus szintén az euklideszi síkon értelmezett minimális Steiner-fát közelíti adott ponthalmaz esetén. Az eddig látott megoldások mindegyike a minimális fesztítőfából indult ki, ez azonban egyetlen lépésében sem használja azt. Ahogyan látni fogjuk, az alapvető ötlete, hogy három pont esetén pontosan tudjuk, hogyan néz ki a minimális

Steiner-fa. Kezdetben tehát választ az algoritmus három pontot, majd egyesével adja hozzá a további pontokat az meglévő fához.

Az algoritmus első lépésében el kell készíteni a bemenő ponthalmaz átlagát. Ezt úgy tehetjük meg, hogy vesszük a pontok x koordinátáinak a számtani közepét, majd az y koordinátáknak a számtani közepét. Azt a pontot nevezzük a ponthalmaz átlagának, melynek koordinátái éppen az imént kiszámított értékek. Ezek után lemérjük az összes pont távolságát az átlagtól, és növekvő sorrendbe rakjuk őket az előbb mért távolság szerint. A következő lépésben vesszük azt a három pontot, melyek az átlaghoz a legközelebb vannak, és a Toricelli pont megszerkesztésével elkészítjük a minimális Steiner-fát erre a három pontra. Ezek után vesszük azt a pontot, amelyik még nem szerepel a fában és a lehető legközelebb van az átlaghoz. Ezt a pontot - nevezzük t-nek - próbáljuk meg a meglévő fához úgy hozzáfűzni, hogy a kapott fa összhossza a lehető legkisebb legyen. Úgy csináljuk, hogy veszünk először egy élt, pontosabban annak a végpontjait. Így a t ponttal együtt kapunk három pontot, amire megszerkesztjük a minimális Steiner-fát lokálisan. Ezt minden lehetséges él végpontjaival megtesszük, és minden próbálkozásnál kiszámítjuk a fa összhosszát. Amelyik fa összhossza így a legkisebb lesz, azzal a fával folytatjuk az eljárást. Minden lépésben mindig a fában még nem szereplő, az átlaghoz legközelebbi pontot fűzzük hozzá a meglévő fához. Ha az összes ponton végigértünk, akkor a kapott fa lesz a közelítő Steiner-fa, az algoritmus kimenete. Most is megfogalmazzuk az algoritmust formálisan, pontosabban.

Az algoritmus:

INPUT: $T = \{t_1, t_2, \dots t_n\}$ ponthalmaz, a fix pontoknak a halmaza.
OUTPUT: Olyan Steiner-fa, melynek összhossza közelíti a minimális Steiner-fa összhosszát.

1 lépés: Legyenek a t_i pont koordinátái x_i és y_i. Jelöljük a ponthalmaz átlagát t'-vel. A t' pont koordinátáit az alábbi képletekkel kapjuk meg: $x' = \frac{1}{n}\sum x_i$ és $y' = \frac{1}{n}\sum y_i$. Ezután számítsuk ki a fix pontok távolságát a t'-től és állítsuk őket távolság szerinti növekvő sorrendbe. Tehát azzal a ponttal kezdődjön a rendezett sorozat, amelyik t'-höz a legközelebb van. A rendezett sorozat elemeit a korábban használt jelölést felülírva jelöljük $t_1, t_2, \dots t_n$-nel.

2. lépés: A t_1, t_2 és t_3 csúcsokhoz szerkesszük meg a minimális Steiner-fát a korábban bemutatott eljárással. Hívjuk az így kapott fát „akutális fának”.

3. lépés: FOR $k = 4, \dots n$ DO

a) „régi fa” := „aktuális fa”; „legjobb fa” := egy ∞ súlyú mesterséges fa.

b) A „régi fa” minden (a, b) élére végezzük el az alábbi műveleteket:

- Helyezzünk el az (a, b) élen egy s Steiner pontot.
- Töröljük az (a, b) élt.
- Vegyük fel az $(a, s), (b, s)$ és (t_k, s) éleket.

- Mozgassuk el az s pontot a síkon úgy, hogy éppen az a, b és t_k pontokhoz tartozó Toricelli pont helyére kerüljön.
- Ha az így kapott fa öszhossza kisebb, mint a „legjobb fa” összhossza, akkor legyen a „legjobb fa” az imént megszerkesztett fa.

c) Legyen az „aktuális fa” a „legjobb fa”.

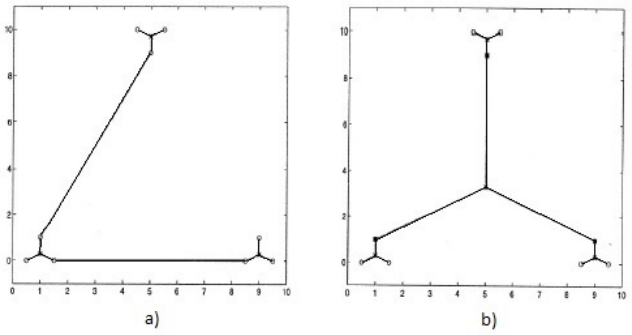
4. lépés: Legyen az algoritmus kimenete az „aktuális fa”.

Vizsgáljuk meg az előbbi algoritmusnál is látott négy pont esetét. Legyen tehát $t_1 = (0{,}0), t_2 = (0{,}1), t_3 = (1{,}3; 1)$ és $t_4 = (1{,}3; 0)$. Mivel a négy pont egy téglalapot alkot, az átlaguk mindegyiktől egyforma távolságra lesz. Így az első három csúcsot tetszés szerint választhatjuk, legyenek ezek például a t_1, t_2 és t_3 pontok. Ehhez a három ponthoz megszerkesztjük a Toricelli pontot (s_1) és elkészítjük a minimális Steiner-fát. Ezután mindössze egyszer fut majd le a harmadik lépésben szereplő ciklus, mégpedig t_4-re. Az esetek végiggondolásával könnyen kapjuk, hogy a legkisebb összhosszú fát akkor tudjuk megszerkeszetni, amikor a (t_3, s_1) élre fut a ciklusmag. Az újabb s_2 Steiner pont így a t_3, s_1 és t_4 csúcsokkal lesz összekötve. Ekkor s_2 természetesen az s_1, t_3 és t_4 pontokhoz tartozó Toricelli pont is egyben.

Minden új csúcs hozzáadásakor egy élt törlünk a korábbi fából és három új élt adunk hozzá, így minden lépésben kettővel növekszik az élszám. Az algoritmus harmadik lépésében a ciklus annyiszor fut le, ahány éle van az aktuális fának. Így először háromszor, majd ötször, hétszer, a legutolsó alkalommal pedig $2n-5$-ször, ha n-nel jelöljük a fix pontok számát. Minden lefutás során az algoritmus megszerkeszti a Toricelli pontot, összeköti a megfelelő csúcsokkal, és meghatározza az így kapott fa összhosszát. Nevezzük ezt lokális optimalizáló eljárásnak. Összesen tehát $3+5+\ldots+2n-5 = (n-3)\frac{2n-2}{2} = (n-3)(n-1)$-szor hívja meg a program ezt az eljárást, így az algoritmus futásideje megegyezik $\theta(n^2)$-szer a lokális optimalizáló eljárás futásidejével. A gyakorlatban ez azt jelenti, hogy ha Matlab programmal teszteljük az algoritmust, akkor 10 pont esetén átlagosan 10 perc, 40 pont esetén 3 óra, 100 pont esetén pedig 1 nap a futásidő.

2.4. Teszt eredmények

Tekintsünk néhány ponthalmazt, és futtassuk le rajtuk először a Steiner pontot beszúró (SPB) algoritmust, majd a növekvő optimalizálási (NO) algoritmust. Mindkét megvalósításnak megvannak a maga előnyei és hátrányai egyaránt. Általánosságban azt tudjuk mondani, hogy az SPB algoritmus mindig sokkal gyorsabb, de a legtöbb esetben az NO algoritmus adja a jobb közelítést. Ugyanakkor nagy méretű, véletlenszerűen generált ponthalmaz esetén az utóbbi nem tudja jobban megközelíteni a minimális Steiner-fát, mint az előbbi gyorsabb algoritmus. A következő szakaszban néhány konkrét példán keresztül vizsgáljuk meg a két közelítést.



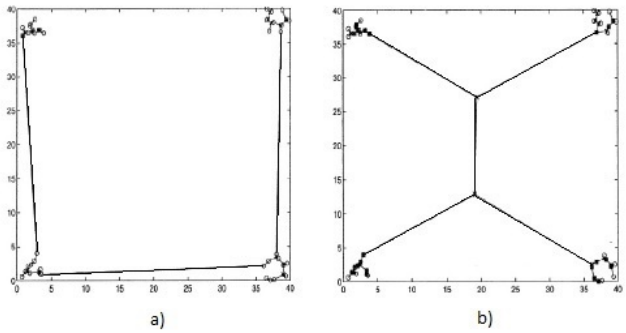
2.2. ábra. Az SPB és az NO algoritmus kimenete az első példára

Először vegyünk fel kilenc pontot a síkon úgy, hogy hármasával egy-egy kicsi háromszöget alkossanak, és minden csoportból egy pontot kivéve egy nagy háromszöget kapjunk. Ahogyan a 2.2/a) ábrán is látjuk, az SPB algoritmus hiába készít el egy olyan fát, ahol minden szög legalább 120 fokos, nem találja meg a nagy háromszög közepén lévő Steiner pontot. Így az NO algoritmus (2.2/b) ábra), amely ezt megtalálja, sokkal jobb közelítést ad. Számszerűsítve ez azt jelenti, hogy a minimális feszítőfa összhosszához képest az SPB algoritmus 3,4% -os javítást tud elérni, míg az NO algoritmus 7,9%-osat. Az NO algoritmus valójában a minimális Steiner-fát adja ebben az esetben.

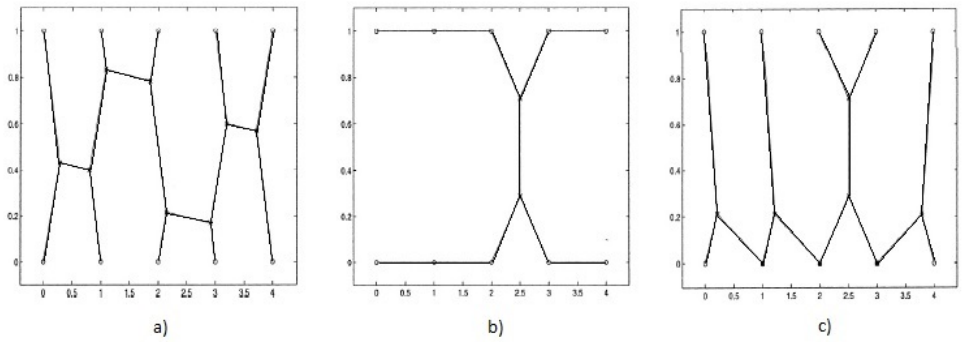
A következő példában az előbbihez hasonló esetet vizsgálunk, melyet a 2.3-as ábra szemléltet. Összesen 40 pontunk van négy tizes csoportban, és a csoportok egy négyzet négy sarkában helyezkednek el. Az SPB közelítő Steiner-fára ismét teljesül, hogy minden szög legalább 120 fokos. Ugyanakkor az algoritmus nem ismeri fel a ponthalmaz meghatározó szerkezetét, azt, hogy a pontok egy négyzetet rajzolnak ki. Az NO eljárás megtalálja és beszúrja a két Steiner pontot a nagy négyzet közepére, ahogyan azt a négy pont eseténél korábban láthattuk. Ezzel 5,4%-kal csökkenti a fa hosszát a minimális feszítőfa hosszához képest, míg az SPB algoritmus csak 0,7%-kal tudja csökkenteni. Fontos azonban kiemelni, hogy az SPB algoritmus futásideje néhány másodperc volt, ellenben az NO algoritmus három órán keresztül dolgozott.

A harmadik bemutatott példa Chung és Graham „létra" tesztje n=10 pontra. A minimális Steiner fát a 2.4/a) ábra mutatja. Az SPB algoritmus a minimális feszítőfa összhosszát 3,0%-kal csökkenti (2.4/b) ábra), az NO algoritmus 5,2%-kal (2.4/c) ábra). A minimális Steiner-fa hossza 7,3%-kal kevesebb, mint a minimális feszítőfa hossza.

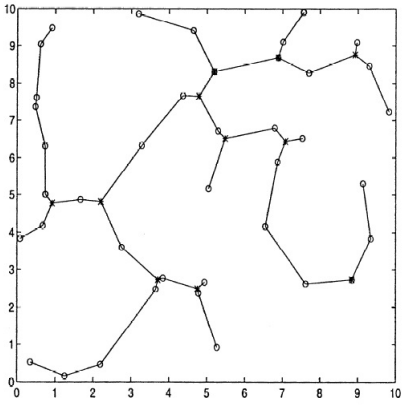
A 2.5-ös ábrán látható negyedik és egyben utolsó példánkban 40 pont szerepel, melyeket véletlenszerűen veszünk fel egy 10×10-es négyzetben. A tapasztalat az, hogy az NO algoritmus nem ad sokkal jobb közelítést, mint az SPB algoritmus. Az tehát az NO algoritmus legnagyobb hátránya, hogy véletlenszerű esetekben, noha a futásideje sokkal hosszabb, mint az SPB algoritmusé, nem tud lényegesen jobb közelítést nyújtani.



2.3. ábra. Az SPB és az NO algoritmus kimenete a második példára



2.4. ábra. A minimális Steiner-fa, valamint az SPB és NO algoritmusok kimenetei a harmadik példára

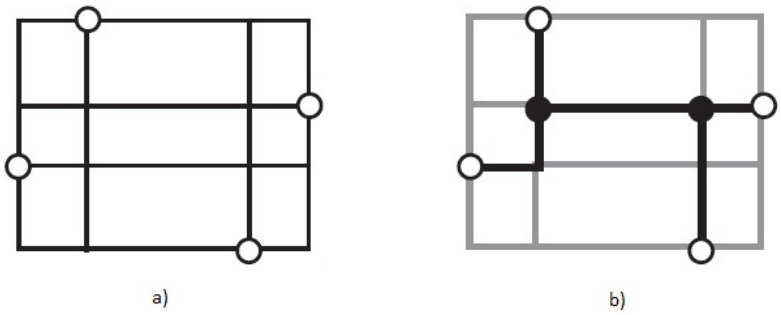


2.5. ábra. Az SPB algoritmus kimenete a negyedik, véletlenszerű esetre

3. fejezet

A taxi probléma

A szakdolgozat utolsó fejezetében térjünk ki a Steiner-fa probléma harmadik változatára, amit taxi problémának vagy Manhattan problémának szoktak hívni. Manhattan utcái olyanok, hogy két utca egymással páhuzamos vagy egymásra merőleges, és az utcák kelet-nyugati vagy észak-déli irányúak. Ha tehát az egyik saroktól el szeretnénk jutni egy másik sarokig, akkor azt úgy tehetjük meg, hogy először például kelet vagy nyugat irányába indulunk úti célunknak megfelelően, majd észak vagy dél felé folytatjuk az utunkat. Ha a két sarkot egy-egy pontnak feleltetjük meg a síkon, akkor az előbbi séta hossza a két pont távolságának felel meg a következő értelemben: két pont távolságán az x és y koordinátáik különbségének összegét értjük. Ha tehát a két pont $a = (x_1, y_1)$ és $b = (x_2, y_2)$, akkor a távolságuk: $dist(a, b) = |x_1 - x_2| + |y_1 - y_2|$. A taxi probléma pontoknak egy halmazából indul ki, melyek a koordinátáikkal vannak megadva. Ezt a ponthalmazt a továbbiakban jelöljük P-vel. A feladat az, hogy egy olyan fát keressünk, melynek a P-ben szereplő pontok csúcsai. A fa egy éle két pontot összekötő vízszintes és függőleges szakaszok olyan váltakozó sorozata lehet, melynek hossza éppen a két csúcs távolságával egyezik meg. A fa egy élének hossza tehát a két végpontjának a távolsága. A cél az, hogy a lehető legrövidebb összhosszú fát adjuk meg. Ebben a formában a feladat nem más, mint minimális feszítőfa keresése. Ha azonban további pontokat vehetünk fel a síkon, melyeket a fa csúcsaiként felhasználhatunk, akkor az új ponthalmaz minimális feszítőfájának összhossza kisebb lehet, mint az eredeti ponthalmaz minimális feszítőfájáé. Az így hozzáadott pontokat Steiner pontoknak nevezzük, és a belőlük álló halmazt S-sel jelöljük. Steiner-fának hívjuk azt a minimális feszítőfát, melynek csúcsai a $P \cup S$ halmaz elemei, minimális Steiner-fának pedig az összes lehetséges Steiner-fa közül a legkisebb összhosszút.



3.1. ábra. Hanan tétele: Van olyan minimális Steiner-fa, melynek Steiner pontjai a Hanan rács pontjai közül kerülnek ki

3.1. Történelmi áttekintés

A következő alfejezetek Gabriel Robins és Alexander Zelikovsky cikkére épülnek [6]. A taxi probléma, vagy más néven Manhattan probléma kutatásának, a közelítő algoritmusok fejlődésének több mérföldköve volt. Most ezek közül említek meg néhányat. Hanan 1966-ban megmutatta, hogy tetszőleges P halmaz esetén van olyan minimális Steiner-fa, melynek Steiner pontja az eredeti pontokon keresztül húzott vízszintes és függőleges vonalak metszéspontjai közül kerülnek ki. A matematikus emlékére ezeket a pontokat a Hanan rács pontjainak hívjuk. A 3.1/a) ábrán a Hanan rácsot láthatjuk, a 3.1/b) ábrán pedig az adott 4 ponthoz tartozó minimális Steiner-fát, ahol a Steiner pontok valóban rácspontok. A későbbiekben bemutatott ismételt 1-Steiner pontot beszúró algoritmus is ezekkel a metszéspontokkal dolgozik.

Annak eldöntése, hogy adott ponthalmaz esetén tetszőleges k számhoz létezik-e olyan Steiner-fa, melynek összhossza legfeljebb k, NP-teljes probléma, ha a Steiner pontokat a Hanan rács pontjai közül választjuk ki. Ezt Garey és Johnson bizonyította 1976-ban, és a tényt röviden úgy fogalmazzuk meg, hogy a taxi probléma NP-teljes feladat.

Jelölje $cost(MST(P))$ a P csúcshalmazhoz tartozó minimális feszítőfa összhosszát, a P-hez tartozó minimális Steiner-fa összhosszát pedig $cost(SMT(P))$. Hwang 1976-ban belátta, hogy a minimális feszítőfa jól közelíti a minimális Steiner-fát, egészen pontosan $\frac{cost(MST(P))}{cost(SMT(P))} \leq \frac{3}{2}$ tetszőleges P ponthalmaz esetén. Ebből az is következik, hogy bármely algoritmus esetén, amely a minimális feszítőfából indul ki, és valamilyen ötlettel annak az összhosszát csökkenti, a közelítő Steiner-fa összhosszának és a minimális Steiner-fa összhosszának az arány legfeljebb 3/2 lehet. Ez a felső korlát a kezdeti ponthalmaztól és minden más paramétertől független, jól kezelhető érték, egy konstans. Éppen ezért a közelítő algoritmusok kedvelt első lépése a minimális feszítőfa keresés.

Hosszú éveken át nyitott kérdésnek számított, hogy létezik-e olyan megoldás, ahol a közelítő

és a minimális Steiner-fa aránya tetszőleges kezdeti ponthalmaz esetén szigorúan kisebb, mint 3/2. Ezt az arányt a közelítés mértékének szokás nevezni. Ha tetszőlegesen sok ponthalmazra egymás után elvégezzük az algoritmus lépéseit, és kiszámítjuk minden esetben a közelítés mértékét, akkor ezeknek az átlagát véve megkapjuk az úgynevezett átlagos közelítést. A tapasztalat azt mutatta, hogy a minimális Steiner-fát közelítő algoritmusok mind hasonló átlagos közelítéssel dolgoznak. Ezért a másik fő törekvés olyan algoritmus megírására irányult, amely az átlagos közelítést tudja javítani. Véletlenszerű, egyenletes ponthalmaz esetén a minimális feszítőfa hosszát $7-9\%$-kal tudták javítani az algoritmusok. 1990-ben Kahng és Robins [7] mutatta meg, hogy bármely mohó, a minimális Steiner-fából kiinduló közelítő algoritmus esetén a közelítés mértéke legrosszabb esetben tetszőlegesen közel lehet 3/2-hez. Azaz megadható olyan csúcshalmaz, mely estén az algoritmus nem tud javítani a minimális feszítőfán, így az összhosszat sem tudja csökkenteni. Ebből az adódik, hogy nincs arra esély, hogy a minimális feszítőfából kiinduló alogritmusoknál a közelítés mértéke 3/2-nél kisebb legyen.

1992-ben Zelikovsky olyan közelítő algoritmust mutatott be a taxi problémára, ahol a közelítő és a minimális Steiner-fa aránya legfeljebb 11/8 lehet. Ezzel egyúttal azt is bizonyította, hogy létezik olyan polinom időben futó algoritmus, ahol a közelítés mértéke határozottan kisebb, mint 3/2. Az imént látottak miatt a tudomány elfordult a minimális feszítőfából kiinduló algoritmusoktól, és új utakat kezdett el keresni. Az első eredmény Kahng és Robins nevéhez fűződik, ők publikálták a következő fejezetben bemutatott ismételt 1-Steiner pontot beszúró algoritmust. A megoldás egyszerű, könnyen érthető és implementálható, továbbá általánosítható több dimenziós esetre és gráfokra is.

3.2. Az ismételt 1-Steiner pontot beszúró algoritmus

Az ismételt 1-Steiner pontot beszúró algoritmus a taxi problémára ad egy olyan Steiner-fát, melynek összhossza közelíti a minimális Steiner-fa összhosszát. Az algoritmusra a későbbiekben az egyszerűség kedvéért I1S algoritmusként fogunk hivatkozni. Mindenekelőtt egy új fogalom bevezetésére lesz szükségünk. Legyen X csúcsoknak egy halmaza. Ekkor $MST(X)$ jelölje azt a minimális feszítőfát, melynek csúcsai X elemei, összhossza pedig legyen $cost(MST(X))$. Az algoritmus lényegében az alábbi fogalomra épül:

$\Delta MST(A,B) = cost(MST(A)) - cost(MST(A \cup B))$, ahol A és B csúcsoknak a halmaza.

$\Delta MST(A,B)$ tehát azt méri, hogy mennyivel nő vagy csökken a minimális feszítőfa összhossza, ha nem csak az A-beli, hanem az $A \cup B$-beli csúcsoknak keressük a minimális feszítőfáját.

Az I1S algoritmus bemenetként megkapja a P csúcshalmazt. Húzzunk minden P-beli csúcson keresztül egy-egy vízszintes és függőleges vonalat. Az így kapott metszéspontok lesznek

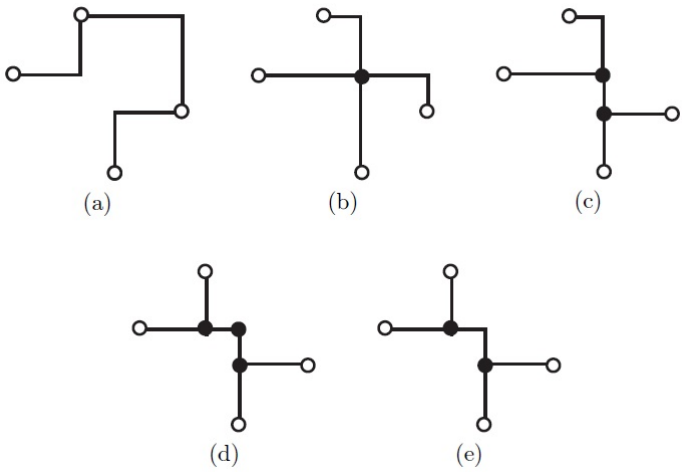
a Steiner pont jelöltjeink, jelöljük a belőlük álló halmazt $H(P)$-vel. Egy $x \in H(P)$ pontot a P-hez tartozó 1-Steiner pontnak nevezünk, ha az alábbi két feltételt teljesíti. Az első feltétel az, hogy $\Delta MST(P, \{x\}) > 0$ legyen, azaz az x pont hozzávételével a minimális feszítőfa súlyát csökkenteni tudjuk. A második kritérium pedig, hogy $\Delta MST(P, \{x\}) \geq \Delta MST(P, \{y\})$ legyen minden $y \in H(P)$-re, azaz a $\Delta MST(P, \{.\})$ mennyiség x-ben vegye fel a maximumát. Ez azt jelenti, hogy az x csúcs választásával tudjuk éppen a legnagyobb mértékben csökkenteni a minimális feszítőfa összhosszát. Az I1S algoritmus tehát az adott P halmazból indul ki. Kezdetben a Steiner pontok halmaza üres, azaz $S = \emptyset$. Minden lépésben egy Steiner pontot adunk a gráfhoz, mégpedig az aktuális ponthalmazhoz tartozó 1-Steiner pontot. Az aktuális ponthalmaz az eredeti csúcsokat és az algoritmus korábbi lépéseiben hozzáadott Steiner pontokat jelenti. Az 1-Steiner pont beszúrása azt eredményezi, hogy minden lépésben a lehető legtöbbel csökkentjük a feszítőfa összhosszát. A lépéseket addig ismételjük, amíg tudunk csökkenteni, azaz találunk az adott ponthalmazhoz 1-Steiner pontot. Ha ilyet már nem találunk, akkor az algoritmus véget ér, a kimenete pedig az éppen aktuális ponthalmaz minimális feszítőfája lesz.

Most pedig vizsgáljuk meg, hogy mit mondhatunk egy Steiner pont fokszámáról. A fokszám nem lehet egy, mivel ha az egyfokú Steiner pontot és a belőle induló élt töröljük, akkor biztosan csökkentjük a fa összhosszát. Ha egy kettő fokú Steiner pontot elhagyunk a gráfból, akkor a Steiner pontból kiinduló két él végpontjait összekötő élt kapunk, amely a Steiner ponton keresztül haladt. Azaz az így elhelyezett Steiner pont semmivel sem csökkentette a fa hosszát. (Sőt, még az is elképzelhető, hogy növelte.) Így azt kapjuk, hogy minden Steiner pont fokszáma legalább három. Az euklideszi Steiner-fa probléma című fejezetben már megmutattuk, hogy ekkor $|P| = n$ esetén a minimális Steiner-fában a Steiner pontok száma legfeljebb $n - 2$ lehet. Azonban előfordulhat, hogy az I1S algoritmus több, mint $(n-2)$-ször tudja csökkenteni a minimális feszítőfa hosszát Steiner pont hozzáadásával. Ezért minden iterációban vizsgáljuk meg, hogy a Steiner pontokból hány él indul ki az aktuális feszítőfában, és tötöljük azokat, melyek fokszáma legfeljebb kettő. A közelítő algoritmus a korábbi jelöléseket megtartva formálisan a következőképpen néz ki:

Az I1S algoritmus:

INPUT: P ponthalmaz
OUTPUT: Olyan Steiner-fa, melynek összhossza közelíti a minimális Steiner-fa összhosszát.

1. lépés: $S := \emptyset$
2. lépés: Amíg $M := \{x \in H(P \cup S) | \Delta MST(P \cup S, \{x\}) > 0\} \neq \emptyset$,
- Keressük meg azt az $x \in M$ pontot, mely $\Delta MST(P \cup S, \{.\})$-t maximalizálja.
- $S := S \cup \{x\}$



3.2. ábra. Az I1S algoritmus működése 4 pont esetén

- Töröljük azokat a csúcsokat S-ből, melyek fokszáma legfeljebb kettő $MST(P \cup S)$-ben.

3. lépés: Az algoritmus kimenete: $MST(P \cup S)$.

A 3.2 ábra az I1S algoritmus működését szemlélteti. Az a) ábra a bemenetként kapott 4 pont minimális feszítőfáját mutatja. Ezután az algoritmus három Steiner pontot szúr be, ezeket a b), c) és d) ábrákon sötét pontok jelzik. A korábbi megállapítás szerint azonban 4 pont esetén a minimális Steiner-fában a Steiner pontok száma legfeljebb 2 lehet. Az egyik Steiner pont fokszáma kettő lesz, így törölhetjük. Ezt az e) ábra is mutatja. Tehát az algoritmus kimenete olyan közelítő Steiner-fa lesz, melynek kettő Steiner pontja van.

Nézzük meg, hogy hány műveletre van szükségünk egy darab 1-Steiner pont megtalálásához. A Steiner pont jelöltek száma minden iteráció során $O(n^2)$, és minden x jelölt esetén meg kell határozni a $P \cup S \cup \{x\}$ halmaz minimális feszítőfáját, majd kiszámítani az összhosszát. A $|P \cup S|+1$ mennyiség n-nek lineáris függvénye, ezért a minimális feszítőfa keresés műveletigénye a Kruskal algoritmussal $O(nlogn)$, így $O(n^2)$ csúcs esetén ez $O(n^3logn)$-es futásidőt eredményez. Emiatt egy darab 1-Steiner pont megtalálásának műveletigénye $O(n^3logn)$ lesz. A Steiner pontok száma n-ben lineáris, így az I1S algoritmus futásideje $O(n^4logn)$. Mivel egy-egy Steiner pont jelölthöz szerkesztett minimális feszítőfák között sokszor nagyon kicsi az eltérés, kidolgozhatók olyan módszerek, melyek ezt felhaználva az algoritmus futásidejét jelentős mértékben csökkentik.

A gyakorlatban véletlenül egyenletesen generált ponthalmaz esetén az iteráció átlagosan $n/2$-szer fut le mindössze. Továbbá az is igaz, hogy az I1S algoritmus legfeljebb 4 pont esetén a minimális Steiner-fát adja. Ez a kijelentés közel sem triviális, más közelítő algoritmusok már 4 pont esetén sem tudják mindig a minimális Steiner-fát meghatározni.

3.3. A több 1-Steiner pontot beszúró változat

Az I1S algoritmus esetén azt állapítottuk meg, hogy egy darab 1-Steiner pont megtalálásának a futásideje $O(n^3 logn)$ lesz. Más módszereket használva, melyekre nem térünk ki, ez valójában $O(n^2)$ művelettel is megvalósítható. Ennek ellenére a feladat implementálása nem egyszerű, a számítógépnek geomteriai szempontból bonyolult munkát kell végeznie. Ez motiválta a következő szakaszban bemutatott több 1-Steiner pontot beszúró változat létrejöttét. Ez a változat az 1-Steiner pontok keresésének számát csökkenti úgy, hogy egy lépésben nem csak egy darab 1-Steiner pontot ad a gráfhoz, hanem annyi "független", amennyit csak lehet. Azt, hogy mit jelent, hogy két 1-Steiner pont független, rövidesen definiálni fogjuk.

A több 1-Steiner pontot beszúró (továbbiakben T1S) algoritmus szintén a P ponthalmazból indul ki. A Steiner pontok halmazát jelöljük S-sel. A P csúcshalmazhoz elkészíti a $H(P)$ halmazt, amely továbbra is a Hanan-rács pontjaiból, azaz a Steiner pont jelöltekből áll. Ezután minden $x \in H(P)$ csúcshoz kiszámítja a $\Delta MST(P, \{x\})$ értéket. Ha ez pozitív, akkor x-et 1-Steiner pontnak hívjuk. Legyen $x, y \in H(P)$. Ekkor azt mondjuk, hogy x és y függetlenek, ha

$$\Delta MST(P, \{x\}) + \Delta MST(P, \{y\}) \leq \Delta MST(P, \{x, y\}),$$

azaz átrendezve:

$$\Delta MST(P, \{x\}) \leq \Delta MST(P, \{x, y\}) - \Delta MST(P, \{y\}).$$

Ezt úgy is megfogalmazhatjuk, hogy x hozzáadása legalább annyival tudja csökkenteni a fa összhosszát akkor, amikor a P halmazhoz az y pontot már hozzáadtuk, mint amikor y-t még nem adtuk P-hez. Mivel x és y szerepe felcserélhető, ugyanez nyilván elmondható y-ról is. Több pont esetére általánosítsuk a fogalmat úgy, hogy egy $z \in H(P)$ csúcs független az S halmaztól, ha $\Delta MST(P \cup S, \{z\}) \geq \Delta MST(P, \{z\})$. A T1S algoritmus minden iteráció során mohó módon a lehető legtöbb 1-Steiner pontot adja a Steiner pontok halmazához úgy, hogy az újonnan hozzáadott pont az aktuális S halmaztól független. Kezdetben S legyen az üres halmaz. Először az algoritmus kiválasztja azt az 1-Steiner pontot, melyre $\Delta MST(P, \{.\})$ maximális, és a Steiner pontok halmazához adja. Utána $\Delta MST(P, \{.\})$ szerinti csökkenő sorrendben halad az 1-Steiner pontokon, és ha S-től függetlent talál, akkor az szintén Steiner pont lesz. Tehát minden ilyen alkalommal az S halmaz egy elemmel bővül. Ha az 1-Steiner pontokon végigér a ciklus, akkor a P halmazt kell frissíteni az S-beli pontok hozzáadásával. Ezután az új P halmazhoz készítsük el a $H(P)$ halmazt, és ismételjük az imént leírtakat. Az algoritmus akkor áll le, amikor nem talál több 1-Steiner pontot. Most pedig fogalmazzuk meg pontosan az algoritmus működését.

A T1S algoritmus:

INPUT: P ponthalmaz

OUTPUT: Olyan Steiner-fa, melynek összhossza közelíti a minimális Steiner-fa összhosszát.

1. lépés: $S := \emptyset$

2. lépés: Amíg $T = \{x \in H(P) | \Delta MST(P, \{x\}) > 0\} \neq \emptyset$,

- $S := \emptyset$

- Menjünk végig T elemein ΔMST szerinti csökkenő sorrendben, és ha $\Delta MST(P \cup S, \{x\}) \geq \Delta MST(P, \{x\})$, akkor $S := S \cup \{x\}$

- $P := P \cup S$

- Töröljük azokat a csúcsokat P-ből, melyek fokszáma legfeljebb kettő $MST(P)$-ben.

3. lépés: Az algoritmus kimenete: $MST(P)$.

A tapasztalat azt mutatja, hogy a pontok számának növelésével nagyon lassan nő az iterációk száma. Például egy 300 pontos input esetén átlagosan 2,5 iterációra van szükség, és soha nem fordult elő még olyan, hogy 5 ismétlésnél több kellett volna. Az algoritmus által végzett, a minimális feszítőfa összhosszához viszonyított javítás legalább 95%-a az első iteráció során történik, míg az első két iteráció során a javítás 99%-a. A T1S algoritmus által a gráfhoz hozzáadott Steiner pontok átlagos száma a bemenő ponthalmaz lineáris függvénye, jellemzően kevesebb, mint a pontok számának a fele.

Az I1S és a T1S algoritmusok hasonló átlagos közelítéssel dolgoznak, a minimális feszítőfa hosszát körülbelül 11%-kal csökkentik. A közelítő Steiner-fa hossza így a minimális Steiner-fa hosszától általában mindössze 0,5%-kal tér el. Sőt, 8 pont esetén a kimenetek 90%-a, 15 pont esetén a fele, 30 pont esetén pedig a negyede maga a minimális Steiner-fa lesz.

Irodalomjegyzék

[1] L. Kou, G. Markowsky and L. Berman, *A Fast Algorithm for Steiner Trees*, Acta Informatica, **15** (1981) 141-145.

[2] Kurt Mehlhorn, *A faster approximation algorithm for the Steiner problem in graphs*, Information Processing Letters, **27** (1988) 125-128.

[3] M.L. Fredman and R.E. Tarjan, *Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms*, IEEE, (1984) 338-346.

[4] Derek R. Dreyer and Michael L. Overton, *Two Heuristics for the Euclidean Steiner Tree Problem*, Journal of Global Optimization, **13** (1998) 95-106.

[5] F.K. Hwang, D.S Richards and P. Winter, *The Steiner Tree Problem*, (1992).

[6] Gabriel Robins and Alexander Zelikovsky, *Minimum Steiner Tree Construction*, The Handbook of Algorithms for VLSI Physical Design Automation, CRC Press, (2009) 487-508.

[7] A. B. Kahng and G. Robins, *A new family of steiner tree heuristics with good performance: The iterated 1-steiner approach.*, Proc. IEEE International Conf. Computer-Aided Design, (1990) 428–431.

[8] *Wikipedia The Free Encyclopedia*