Bevezetés

Kiszámítási probléma: I bemenetre $f(I)$ kimenet.

Eldöntési probléma: $f(I)$ = igen vagy nem.

Optimalizálási probléma: I bemenetre létezik egy X_I, a lehetséges kimenetek halmaza, ami közül ki kell választani egyet, például összefüggő élsúlyozott gráfban kell keresni minimális összsúlyú feszítőfát. Ekkor a gráf I, X_I pedig a lehetséges feszítőfák halmaza. Egy $c(X)$ **célfüggvény** mondja meg, hogy mennyire közel vagyunk a megoldáshoz, itt például a feszítőfa súlya. Azt az X^*-ot keressük, amire c minimális vagy maximális, ez a feladat természetétől függ. A megoldást **optimum**nak hívjuk.

Másik példa egy gráf kromatikus száma, itt I egy gráf, X_I a gráf lehetséges színezéseinek halmaza, $c(x)$ egy konkrét számot rendel az x színezéshez, a színek számát. Ebből minimumot szoktunk keresni.

Közelítő algoritmus: Nem minden feladat oldható meg számítógéppel, ilyen például a megállási probléma (hogy egy kód végtelen ciklusba kerül-e). A tárgy olyanokkal foglalkozik, amik azért megoldhatók, de nem áll rendelkezésre gyors és pontos megoldás, ezért beáldozzuk a pontosságot a sebességért.

Bemenet mérete: n számjegyű egésznek $floor(\log n) + 1$, mert 2-es a számrendszer. Mivel a $floor$ és a $+ 1$ sokat nem számít, ezért elhagyhatók. A tárgy során $\log = \log_2$.

Halmazok kódolása: Ha van például egy $H = \{1, 2, 3, 4\}$ halmazunk, aminek $H' = \{1, 3\}$ részhalmaza, azt például az $\{1, 0, 1, 0\}$ **karakterisztikus vektor** kódolja: amelyik elem benne van az eredeti halmazból a részhalmazban, az 1, a többi 0. Gráfok kódolásához jó például a szomszédossági mátrix.

Gyorsnak a **polinomiális** lépésszámú algoritmusokat hívjuk, a legrosszabb eseti lépésszámot nézve. Egy f függvény futása polinomiális, ha léteznek olyan c_1 és c_2 konstansok, hogy a futásideje $< c_1 * n^{c_2}$. Becsülhetjük felülről más polinommal, és a polinomokat néha csúnyán írják fel, pl. $3n\sqrt{n} = 3n^{\frac{3}{2}}$, amit még azzal is becsülhetünk felülről, hogy $\sqrt{n} < n$, ezért $3n\sqrt{n} < 3n^2$.

- $1 < \log n$
- $\log n < n$
- $2^{\log n} = n$
- $n < n \log n$
- $n \log n < n^2$ (de csak ha n hatványa nagyobb, mint 1)
- $n^2 < 2^n$ (ezt úgy lehet bizonyítani, hogy a hányadosuk végtelen, azt nem becsülheti felülről semmilyen konstans)
- $2^n < n!$

- $2^n < n^n$

Gyakorlati példák:

- $\frac{n^3}{3} + \frac{n^2}{2} \leq n^3 + n^3 = 2n^3$ ⇒ polinomiális
- $n \log n + 3\sqrt{n} \leq n^2 + 3n^2 = 4n^2$ ⇒ polinomiális
- $3^{\log n} \leq 4^{\log n} = 2^{2 \log n} = \left(2^{\log n}\right)^2 = n^2$ ⇒ polinomiális
- $\sqrt{n}^{\log n} = n^{\frac{1}{2}\log n}$, ami nehéz eset, viszont lehet ilyet:
  - $n^{\frac{1}{2}\log n} \leq c_1 \cdot n^{c_2}$
  - $n^{\frac{1}{2}\log n - c_2} \leq c_1$ ⇒ nem lehet polinomiális
- $log\left(n^{n^2}\right) = n^2 \cdot \log n \leq n^3$ ⇒ polinomiális
- $log\left(n^{n^n}\right) = n^n \cdot \log n \geq 2^n$ ⇒ nem polinomiális

Egy algoritmus attól lesz valójában polinomiális időben megoldható, ha ***a lépésszáma a bemenet méretében polinomiális függvény***. Ha van egy I input és az algoritmus mérete n, akkor az algoritmus $f(n)$ lépésszáma kell legyen polinomiális.

Példa: az input a és b, a feladat egyszerű hatványozás, a^b. Ekkor a bemenet mérete $\log a + \log b$, a kiírandó számjegyek száma $\log a^b = b * \log a$, ami a bemenet méretében már exponenciális (a bemenet logaritmikus, a futásidő lineáris, logból lin-t csak exponenciális függvény csinálhat), vagyis az eredmény kiírása sem fut polinomiális időben, nemhogy a probléma. Általában nem ilyen könnyű megmondani egy feladatról, hogy nem megoldható polinomiális időben.

P: polinom időben megoldható eldöntési problémák.

NP: olyan eldöntési problémák, amikre létezik polinom méretű és polinom időben eldönthető tanú (megoldás bizonyítéka) az igen válaszra, pl. Hamilton-kör keresésére konkrét tanú egy Hamilton-kör, de a nem válaszra már nincs ilyen tanú.

coNP: ugyanaz, csak nemleges válaszra. $P \subseteq NP$ és $P \subseteq coNP$, hiszen ha mindenre van polinomiális megoldás, az igen/nem válasz igazolására is lesz.

Polinomiális visszavezetés: B-t polinomiálisan visszavezetjük A-ra: adunk egy polinomiális algoritmust a B megoldására úgy, hogy egy A-t megoldó szubrutint 1 lépésként meg tudunk hívni. "B nem nehezebb, mint A."

NP-nehéz probléma: olyan, amire minden NP-beli probléma visszavezethető polinom időben, például a Hamilton-kör problémája vagy a Boole-formulák kielégíthetősége. Ha egy ilyen problémát meg tudunk oldani polinomiális időben, akkor mindent.

NP-teljes probléma: NP-beli és NP-nehéz.

NP-nehéz problémák megoldásának módja:

1. Nem polinomiális, de viszonylag gyors algoritmust választunk, pl. simplex a lineáris programozásra.
2. Keresünk speciális eseteket, ami polinomiális.
3. Közelítő algoritmusok használata.
4. Heurisztikák használata (olyan, majdnem közelítő algoritmusok, amik nem garantáltan polinomiális időben futnak, pl. nem tudjuk eldönteni a futási időt, de a gyakorlatban tűrhető időben futnak).

Problémák a tárgyon:

1. A leghosszabb út (speciális esete a Hamilton-út) problémája NP-nehéz, de körmentes irányított gráfban van polinom idejű megoldása.
2. Erőforrások ütemezése, mert ez valójában gráf színezése, de intervallumgráfban polinomiális.
3. Maximális független ponthalmaz keresése (semelyik kettő közt nincs él, ebből a legtöbbet válasszuk ki), páros gráfban viszont P-beli.
4. Az élszínezés NP-nehéz, de párosban ez is P-beli.

Optimális élszínezés meghatározása

König tétele: páros gráfnál az élkromatikus szám (legkevesebb ennyi színből lehet megoldani, hogy két, azonos csúcsba futó él ne legyen azonos színű) egyenlő a maximális fokszámmal: $x_e(G) = \Delta(G)$. Ismétlés: algoritmus eredményeként nem egy élkromatikus számot várunk, hanem egy tanút (megoldást), ami bizonyítja, hogy lehetséges annyi színnel kiszínezni.

Minden csúcsnak legyen egy listája az ott még használható $\Delta(G)$ darab színből, vagyis hogy milyen színű él nem indul ki belőle. Élenként vizsgálódunk, amikor egy él két csúcsának van közös eleme, olyan színű legyen az él. Ha nincs közös elem, akkor az aktuális színezést kell megbolygatni:

1. Veszünk egy-egy színt, ami még a két csúcs listáján van, ezek olyan színeket jelölnek, amilyen él a másik csúcsba már befut.
2. Vesszük azt a részgráfot, ami az összes élet tartalmazza azzal a két színnel.
3. Vesszük azt a kis részt, ami az egyik csúcsot tartalmazza, abban felcseréljük a színeket. Ha ezt megtesszük, garantált, hogy mindkét csúcsból csak az egyik színű él fog kifutni, hiszen ahhoz, hogy ez visszahasson, páros lépésű út kellene egy páros gráf két oldala közt, ami nincs.
4. Most már kiszínezhetjük a felszabadult színnel az élet.

Additív hibával közelítő algoritmusok

Olyan megoldást akarunk találni, ami legfeljebb egy konstanssal tér el az optimumtól. Egy algoritmus C additív hibától eltekintve jól oldja meg a problémát, ha minden inputra ad egy $X^* \in X_I$-t, amire teljesül, hogy minden inputra ad egy X^* megoldást, amire teljesül, hogy $c(X^*) \leq min(c(x) + C)$, ha minimalizálási a probléma, vagy $c(X^*) \leq max(c(x) - C)$, ha maximalizálási a probléma, és polinomiális időben fut.

- Élszínezés egyszerű gráfokon

    - **Vizing tétele:** $\Delta(G) \leq x_e(G) \leq \Delta(G) + 1$, tehát bármilyen algoritmus, ami kiad egy színezést, 1 hibával közelít.
- Csúcsszínezés egyszerű gráfokon
    - A négyszíntétel bizonyításából is nyerhető polinomiális idejű algoritmus, ami 1 additív hibával közelít, mert 4 színt használ, 1 szín csak az életlen gráfhoz elég, 2 a pároshoz, ezeket kiszűrjük, maradnak a 3 vagy 4 élkromatikus számú gráfok.
- Leghosszabb kör keresése
    - NP-nehéz, mert tartalmazza speciális esetként a Hamilton-kört. Erre semmilyen C konstansra nem létezik additív hibával közelítő algoritmus.

Multiplikatív hibával közelítő (k-approximációs) algoritmusok

Ezúttal olyan $X^* \in X_I$-t keresünk, amire teljesül, hogy minden input esetén talál olyan X^* megoldást, hogy $c(X^*) \leq k * min/max(c(x))$, és polinomiális időben fut. Ekkor azt mondjuk, hogy k multiplikatív hibától eltekintve helyesen oldja meg a feladatot. A k értékét approximációs faktornak hívjuk.

Maximális páros részgráf

Olyan része egy gráfnak, amiben lehetséges két olyan csoportot kialakítani, ahol csak csoportok közt halad él. Ez a probléma NP-nehéz. Az élek számára maximalizálunk, hiszen csúcsból mindet bevehetjük, mert az élek nélkül bármilyen felosztásban páros. Átfogalmazva a feladat: találjuk meg azt a legnagyobb kettéosztást, ahol az élek csak a két osztály közt haladnak.

1. Tetszőlegesen kettéosztjuk a csúcsokat.
2. Keresünk olyan csúcsot, amiből több él megy saját osztályba, mint a másikba. Az ilyet átpakoljuk a másik osztályba. Ha nincs ilyen, vége az algoritmusnak.
3. Goto 2.

Az eljárás során mindig nő a keresztbe menő élek száma, ezért garantáltan leáll, ráadásul polinomiális időben. Ez egy 2-approximáció, vagyis legalább fele akkora részgráfot talál, mint ami a helyes válasz.

Egy jobb opció, hogy kezdetben üres a két halmaz, és egyesével helyezzük el őket, rendre arra az oldalra, ahol több keresztbe futó élet eredményeznek, de ez is 2-approximáció.

Minimális lefogó ponthalmaz

Ez a feladat is NP-nehéz, mert a maximális független ponthalmaz problémája ekvivalens vele. Ami viszont fura, hogy a minimális lefogóra van 2-approximáció, viszont a maximális függetlenre még approximációt adni is NP-nehéz feladat.

Az approximáció: veszünk egy maximális párosítást, ami polinom időben megy általános gráfban is. Ez a legnagyobb független ponthalmaz, és mivel párosítás, a legnagyobb független élhalmaz kétszerese (egy élhez két pont tartozik), amiről belátható, hogy maximum kétszer akkora, mint az optimum, mert a Gallai-tétel szerint a legnagyobb független élhalmaz kisebb, mint a legkisebb lefogó ponthalmaz, jelen esetben mindkettőhöz egy kettes szorzó társul.

Másik módszer: nem bővíthető párosítást keresünk (olyan párosítás, amibe nem kerülhet új csúcs anélkül, hogy a párosítás sérülne, de nem feltétlenül maximális párosítás), és annak vesszük a végpontjait. Ez is 2-approximáció, csak kisebb eredményt adhat.

Súlyozott halmazfedés probléma

Adott egy U alaphalmaz, ami n elemű, és ennek R_i részhalmazai, amik uniója lefedi U-t. Adott még egy c függvény, ami minden R_i-hez súlyt rendel. A cél, hogy minimális összsúllyal fedjük le U-t. Ez egy NP-nehéz feladat, mert a minimális lefogó ponthalmaz egy speciális esete. Annyira NP-nehéz, hogy még k-approximációs algoritmus sincs hozzá. Olyan approximációs faktor már van, ami valamennyire függ a bemenettől. Legyen k a max halmazméret R-ben, erre adunk egy H_k-approximációt, ahol $ln\, k \leq H_k \leq ln\, k + 1$. Algoritmus: az első körben válasszuk ki azt az S halmazt, aminél $\frac{c(S)}{|S|}$ minimális. Itt $|S|$ nem a halmaz számossága, hanem hogy hány elemet fedhetünk le vele. Ezek után legyen C a már fedettek halmaza, innentől pedig a $\frac{c(S)}{|S \setminus C|}$ képlet játszik,vagyis a legtöbb új elemet lefedő halmaz kerül a végeredménybe. Igazából általános felírásnak is jó, hiszen kezdetben C üres halmaz. Mivel minden lépésben garantáltan adunk hozzá új elemet, az algoritmus polinomiális időben fut.

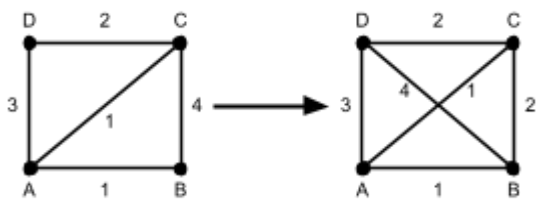
Steiner-fa probléma

Adott egy G összefüggő gráf, aminek csúcsait Steiner-pontokra (S) és terminálokra (T) osztjuk. Steiner-fának fogjuk hívni, ami T összes csúcsát tartalmazza, S-ből mindegy, hogy mennyit, azok csak segédpontok (Steiner-pontok). Adott még egy $c(e)$ súlyfüggvény, ami az élekhez rendel súlyokat/költségeket. Olyan Steiner-fát keresünk, amiben az összköltség minimális. Ez a feladat NP-nehéz, de van rá 2-approximáció.

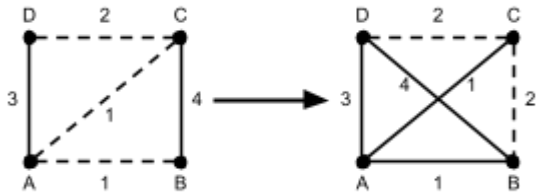
Metrikus Steiner-fa probléma

G ekkor teljes gráf, ugyanúgy S-re és T-re osztva, $c(e)$ költségű élekkel. A költségek teljesítik a háromszög-egyenlőtlenséget, vagyis x, y, és z pontokra, amik közt él halad, $c(xz) \leq c(xy) + c(yz)$. Ha erre tudunk adni k-approximációt, akkor a fő problémára is létezik ilyen.

Metrizálás: ha G gráfban adott a Steiner-fa probléma, létre tudjuk hozni a G' metrikus verziót, amiben a súlyfüggvény legyen c', ahol $c'(xy) =$ x-ből y-ba vezető legolcsóbb/legrövidebb út költsége G-ben c szerint.Ez azt jelenti, hogy ha volt köztük G-ben él, akkor $c(xy) \geq c'(xy)$. Nemcsak átsúlyozzuk az éleket, de kitöltjük teljes gráfra is, ugyanezzel a logikával, a csúcsok közti F-beli legrövidebb út lesz a súlyuk. Egy példa egy G gráf (bal oldalt) metrizálására G'-be (jobb):



Itt $S = \{A\}$ és $T = \{B, C, D\}$, a Steiner-fáik pedig:



Úgy jött ki, hogy T-t bejelöltük G'-ben (BC és CD élekkel), más pont nem is kell, minden le van fedve, majd G-ben a 2 értékű BC élet az azt 2 értékűre csökkentő úttal helyettesítettük.

Miután G'-ben van egy F' Steiner-fa, szeretnénk G-ben is megkapni, de nem feltétlenül szerepel minden éle benne. Ha xy éle F'-nek, akkor vegyük be F''-be az x és y közti legrövidebb utat. Ekkor egy élen többször is áthaladhatnánk, amitől már nem lenne fa, így el kell dobjuk a többszörösen szereplő éleket. Ha ez sem elég, vesszük az így előállt eredmény feszítőfáját (ami mondjuk alapból eldobná a többszörös éleket). A végeredményt F'''-ként jelöljük, és ez F''-nek egy feszítőfája, ezért $c(F''') \leq c(F'') = c'(F') \leq k * OPT$.

Összefoglalva: approximációs algoritmus a metrikus Steiner-fa problémára, hogy a T által feszített részgráfban veszünk egy minimális feszítőfát. Ha kapunk egy nem metrikus problémát, akkor metrizálás után csináljuk ugyanezt, változatlan S és T felosztással.

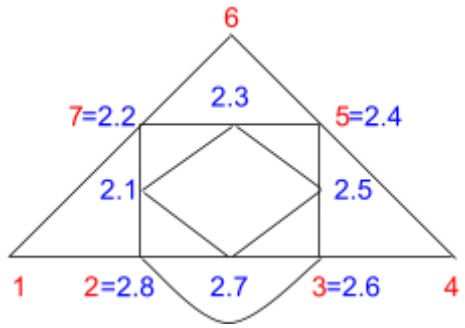
Utazóügynök probléma

Meg kell látogatnia bizonyos városokat, amik között adottak a költségek. Mindegyiket egyszer látogathatja meg, de az összeset meg kell, a legolcsóbb bejárást keressük, vagyis ez egy súlyozott Hamilton-kör probléma, ami NP-nehéz. Adott egy G gráf, $c(e)$ élsúlyok, minimális összköltségű Hamilton-kört (minden csúcsot érint) keresünk benne. Semmilyen k konstans esetén nem létezik hozzá k-approximációja.

Metrikus utazóügynök probléma

A feladatkiíráson felül az élsúlyok teljesítik a háromszög-egyenlőtlenséget, erre már létezik 2-approximáció.

Egy polinomiális algoritmus Euler-körséta (minden élet érint) legyártására:

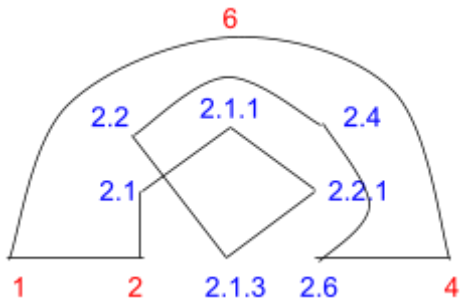


Először bejárunk valamilyen kört (1-től 7-ig, majd vissza 1-be), majd végignézzük a bejárt csúcsokat, hogy valamelyikből indulhat-e újabb bármilyen kör, azt befűzzük az eddigi körbe. A 2-ből indul, legyen 2.1-2.8-ig be is járjuk. Addig ismételjük, amíg minden élet be nem jártunk, például a 2.1-ből kiindulva lehet egy 2.1.1-2.1.4 kört futni a belső rombusz éleivel.

Algoritmus az utazóügynök-problémához:

1. Veszünk egy minimális költségű feszítőfát: F
2. F éleit megduplázzuk, mert ha minden csúcs fokszáma páros, akkor lesz benne Euler-körséta, legyen az eredmény jele F'
3. Keresünk F'-ben Euler-körsétát: F''
4. F''-ből legyártunk egy Hamilton-kört: F'''

Az Euler-körséták után úgy hozunk létre Hamilton-kört a kiteljesített gráfokban, hogy sorban összekötjük a bejárt pontokat, a beszúrt köröket is beleértve, például ha van 1, 2, 3 pontunk, a 2-ből pedig kiindul 2.1, 2.2, 2.3, akkor a végeredmény 1, 2, 2.2, 2.3, 3, mert csak azokat vesszük be, amik még nem voltak. Az előző példa Hamilton-köre így néz ki:



Ez a Hamilton-kör az algoritmus kimenete. Azért jó a metrikusság, vagyis a háromszögegyenlőtlenség, mert $c(F''') \leq c(F'') = c(F') = 2c(F)$, ebből következik a $k = 2$.

A 2-es lépés tovább optimalizálható, elkerülhető az élek megduplázása. A páratlan csúcsok által feszített részgráfban veszünk egy teljes párosítást, és ha ezt rátesszük a részfára, akkor minden csúcs élszáma pont 1-gyel fog nőni, vagyis páros lesz, ezután már mehet a 3. lépés. Az új algoritmustól az approximációs faktor már $\frac{3}{2}$, és úgy hívjuk, hogy Christofides-algoritmus.