Rendszeroptimalizálás tételek 2018

Tételek

1. Az optimális hozzárendelés problémája, Egerváry algoritmusa.

Magyar módszer

Algoritmus

1. Adott páros gráf és egy párosítás (üres is jó)
2. Keres javító utat
3. Ha nincs, a jelenlegi párosítás maximális
4. Javító út mentén az élek szerepét felcseréjük (párosított <> párosítatlan)
5. Go to 2

Alternáló út

Párosítatlan csúcsból induló út amiben párosított és párosítatlan élek felváltva szerepelnek.

Javító út

Párosítatlan csúcsba vezető alternáló út.

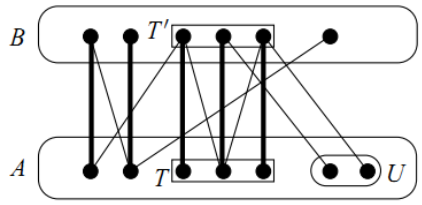
Javító ú keresése

Módosított BFS algoritmus:

- Csak párosítatlan csúcsból indítható.
- Párosított és párosítatlan éleket felváltva választ.

Helyesség bizonyítás

Ha leállt az algoritmus az alábbi helyzet alakul ki:



A, B	Páros gráf csúcsosztályai
M	A jelenlegi párosítás
U	A\M (A csúcsok közül az M által fedetlenek)
T’	U-ból alternáló úton elérhető B csúcsok
T	T’ párosításai

T’ u (A \ (T u U)) lefogó csúcshalmaz, hisz:

- (T u U)-ból induló élek T’-be kell, hogy menjenek (különben alternáló úton elérhető).
- A csúcshalmaz többi csúcsa pedig a lefogó csúcshalmaz része.

|T’ u (A \ (T u U))| = |M|, hisz:

Minden párosításbeli élnek pontosan egy csúcsát tartalmazza.

König tétele szerint páros gráfban a maximális párosítás mérete megegyezik a minimális lefogó csúcshalmazéval. Mivel találtunk |M| méretű lefogó csúcshalmazt, ezért M maximális.

Optimális hozzárendelés

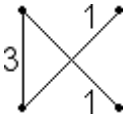
Feladat

Adott $G = (A, B; E)$ páros gráf és w: E $\rightarrow$ **R** élsúly-függvény.

Keressük maximális összsúlyú *M* teljes párosítást: $\max\{\sum_{e \in M} w(e)\}$.

Maximális összsúlyú teljes párosítás vagy maximális összsúlyú párosítás

A maximális összsúlyú párosítás és a maximális összsúlyú teljes párosítás feladat nem ekvivalens.



Pl. a gráfban az első esetben 3 az optimum, a második esetben 2.

Maximális összsúlyú párosítás visszavezetése a maximális összsúlyú teljes párosításra

- Az egyik csúcshalmazt kibővítjük, hogy a kettő elemszáma megegyezen
- A hiányzó éleket 0 súllyal felvesszük
- Negatív súlyú éleket 0 súlyúvá írjuk át

Visszavezetés helyességének bizonyítása

Legyen M1' a maximális teljes párosítás a transzformált gráfon.
Legyen M1 az eredeti gráfon egy párosítás, amit a 0 súlyú élek lehagyásával kaptunk.
(Ez nyilván súlyban megegyezik M1'-vel és csak az eredeti gráf éleit tartalmazza, hisz minden felvett vagy módosított él 0 súlyú, a párosítás tulajdonságát se veszti el, hisz csak éleket hagyunk el.)
Indirekt: tfh létezik M2, amely súlyban nagyobb M1.nél.
Ekkor viszont a transzformált gráfon létezik egy M2' teljes párosítás, amely nagyobb egyenlő súlyú, mint M2 (negatív élek helyett 0 súlyú élt választunk és a fedetlen csúcsokat 0 élekkel veszük be a párosításba).
w(M2') >= w(M2) > w(M1) = w(M1'), de M1' maximális volt, tehát ellentmondásba ütköztunk, nem létezhet M2 párosítás.

Címkézés

Definíció

$G = (A, B; E)$ páros gráf.
c: $(\mathrm{A} \cup \mathrm{B}) \rightarrow \mathbf{R}$ címkézés (csúcs súlyfüggény), ha minden $e = \{x, y\}$ élre $c(x) + c(y) \geq w(e)$.

Címkézés és teljes párosítás súlyának kapcsolata

Lemma

$$\sum_{e=\{x,y\} \in M} w(e) \leq \sum_{v \in A \cup B} c(v)$$

Ha w súly függvényhez képest c címkézés és M egy párosítás.

Bizonyítás

Legyen *M* egy tetszőleges teljes párosítás. Mivel *M* élei minden pontot egyszer fognak le, ezért

$$\sum_{e=\{x,y\} \in M} w(e) \leq \sum_{e=\{x,y\} \in M} c(x) + c(y) = \sum_{v \in A \cup B} c(v)$$

* A címkézési tulajdonság miatt.

** Mivel M teljes, ezért minden csúcsot tartalmaz pontosan egyszer.

Éles címkézés és teljes párosítás súlyának kapcsolata

Lemma

$G = (A, B; E)$ páros gráfban *w* élsúly-függvény, *c* címkézés, *M* teljes párosítás és minden $e = \{x, y\} \in M$ élre $c(x) + c(y) = w(e)$ teljesül, akkor *M* maximális összsúlyú.

Bizonyítás

$$\sum_{e=\{x,y\}\in M} w(e) = \sum_{e=\{x,y\}\in M} c(x) + c(y) = \sum_{v\in A\cup B} c(v)$$

* Az éles címkézési tulajdonság miatt.

** Mivel M teljes, ezért minden csúcsot tartalmaz pontosan egyszer.

Egerváry algoritmus

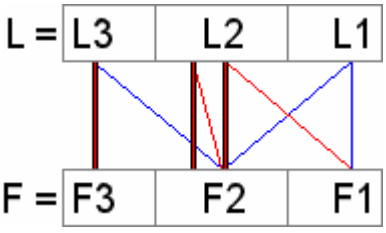
Előfeltétel

A gráfokat kiegészítjük úgy, hogy létezzen bennük teljes párosítás.
(Nem feltétlenül kell, hogy teljes gráf legyen.)

Algoritmus

Egy (F,L;E) práos gráfon számon tart egy c címkézést és egy M párosítást, a kettőt addig javítja míg a párosításon éles lesz a címkézés, ezzel garantálva, hogy maximális élsúlyú.

1. $M = \emptyset$;
   $c = \max_{v \in e, e \in E} w(e), ha\ v \in F$
   $c = 0, ha\ v\ \in L$
2. $PIROS = \{\, (x, y) \in E \mid c(x)\ + c(y) = w\big((x, y)\big) \,\}$
3. M = Magyar módszer M-ből kiinudlva a PIROS részgráfon
4. M teljes párosítás eredeti gráfon => M maximális
5. Ha M még nem TP, akkor élesítjük c címkézést



F1 M álal nem fedett
L2 F1-ből piros alternáló úton elérhető
F2 L2 párja M-ben
L1 M általá nem fedett
L3 A maradék
F3 A maradék

$$\delta = \min\{c(x) + c(y) - w(\{x, y\}) | x \in F_1 \cup F_2,\ y \in L_1 \cup L_3\}$$

$$c(v) = \begin{cases} c(v) - \delta, & \text{ha } v \in F_1 \cup F_2 \\ c(v) + \delta, & \text{ha } v \in L_2 \\ c(v), & \text{különben} \end{cases}$$

(Minden piros él F1 U F2 és L2 között megy, szeretnénk új éleket bevenni, hogy bővülhessen a párosítás, ezért az F1 U F2 és L1 U L3 közöttiek közül a legpirosközelebbit bevesszük az F1 U F2 címkézés csökkentésével, de L2-ben pedig növeüljük, hogy ott se romoljon el a piros tulajodnság.)

6. Go to 1.

Bizonyítás

Létezik-e delta:
Azaz létezik-e L1 u L3 és F1 u F2 között él, a válasz igenlő, hisz a Hall-feltétel miatt egyébbként TP se létezhetne.

Kell még, hogy delta értéke nem 0, hisz különben nem javulna a címkézés, de ez is igaz, hisz csak akkor lehetne 0, ha piros él mentén találtuk meg a minimumot, de F1 u F2 összes piros szomszédja L2-ben kell legyen, különben létezne javító út.

c címkézés marad:
$c(x) + c(y)$ változása:

	$F_1 \cup F_2$	F_3
L_2	0 (- δ+ δ)	$+\delta$
$L_1 \cup L_3$	$-\delta$	0

A címke növekedése a címkézési tulajdonságot nem rontja el (nagyobb egyenlőség).
A L1 u L3 és F1 u F2 között pedig úgy választottuk meg δ-t, hogy minimális lesz, ezért sehol nem fog elromlani a címkézési tulajdonság, de legalább egy helyen élesülni fog.

Eltűnő piros élek:
Csak a L2 és F3 közötti élek esetén nő a címkézés, itt elromolhat a piros tulajdonság, ám ezek nem lehetnek részei az M párosításnak, illetve az L2-beli csúcsok továbbra is elérhetőek maradnak F1-ből piros alternáló úton.

Keletkezik egy új piros él F1 u F2 és L1 u L3 között:
F1-ből elérhető alternáló úton F2, ezért bármelyikből is indul az újonnan piros él a másik vége alternáló úton elérhető lesz, így L2 mérete nőni fog.

O(n) iteráció után (ha még nem nőtt) már minden L-beli csúcs elérhető lesz alternáló úton, azaz ha még M nem teljes és létezik teljes párosítás, akkor nőni fog a párosítás mérete.
$O(n^2)$ iteráció után n-szer nőtt a párosítás azaz teljesnek kell lennie.
Egy iteráció $\delta, F3 \text{ és } F2$ kereséséből áll, ami O(e).

Tehát az algoritmus $O(n^2e)$ lépés után előállít egy teljes párosítást, amin éles a címkézés, azaz maximális összsúlyú.

2. A lineáris programozás alapfeladata, annak mátrixos alakja. Kétváltozós lineáris programozási feladatok grafikus megoldása. Farkas-lemma a lineáris egyenlőtlenségrendszerek megoldhatóságára („első alak”), csak a szükségesség (a „könnyű irány”) bizonyításával.

Lineáris programozás

Alapfeladat

A lineáris programozás alapfeladata: egy linerás egyenlőtlenségrendszer megoldásai közül kiválasztani azt, amely egy szintén lineáris célfüggvény szerint optimális.

Formális definició

$\max\{cx: Ax \leq b\}$
$A \in \mathbb{R}^{mxn}, x \in \mathbb{R}^n, b \in \mathbb{R}^m, c \in \mathbb{R}^n$

Átalakítások

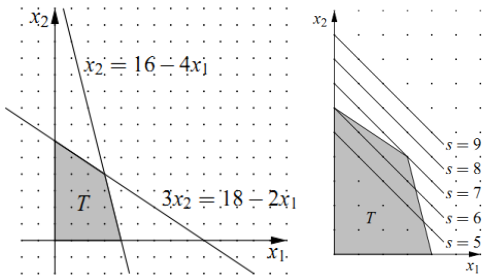
- $\alpha x \geq \beta$ helyett $-\alpha x \leq -\beta$
- $\alpha x = \beta$ helyett $\alpha x \leq \beta$ és $-\alpha x \leq -\beta$
- $\min\{cx: Ax \leq b\}$ helyett $\max\{(-c)x: Ax \leq b\}$ (aztán ellentett-vétel a végleges megoldáshoz)

Kérdések

1. Van-e Ax≤b-nek megoldása?
2. A megoldások halmazán cx korlátos-e?
3. Melyik x-re maximális cx?

A kétváltozós feladat grafikus megoldása

- Minden $\alpha x \leq \beta$ zárt félsíkot határoz meg, határolója $\alpha x = \beta$ egyenes.
- A felsíkok metszetében léteznek a megoldások (bal ábra).
- A célfüggvényt különböző értékekre felrajzolva kiválaszthatjuk azt a célfüggvény – sokszög metszéspontot, amely optimális (jobb ábra; s=7, x_1=3, x_2=4 a megoldás).
  A célfüggvény meghatározza a meredekséget, egy egyenesen az azonos értékű megoldások lesznek. Azaz addig próbáljuk „felfelé” csúsztatni a célfüggvény egyenest, amíg már nem metszi a megoldások halmazát, itt találjuk meg az optimális megoldásokat.



Farkas lemma 1.

Tétel

Pontosan az egyiknek van megoldása:

(1) $Ax \leq b$

(2) $yA = 0; y \geq 0;\ yb < 0$

Bizonyítás

Kettőnek egyszerre nincs megoldása, mert: $0 = 0x = (yA)x = y(Ax) \leq yb < 0.$
(1) nem megoldható => (2) igen
$\nexists x: Ax \leq b \ \Rightarrow \exists y': y'(A|b) = (0..0| < 0), y \geq 0$

Vizsgáljuk a $C = \{z \in \mathbb{R}^{n+1} : z = y(A|b), y \geq 0\}$.
(A cél ekkor belátni, hogy $\nexists x: Ax \leq b \ \Rightarrow \exists z: z = (0..0| < 0), z \in C$)
C tulajdonságai, ha $z_1, z_2 \in C, \lambda > 0$, azaz $z_i \in C \ \Rightarrow \ \exists: z_i = y_i(A|b), y_i \geq 0$

1. $z_1 + z_2 \in C$
   $z_1 + z_2 = y_1(A|b) + \ y_2(A|b) = (y_1 + \ y_2)(A|b), (y_1 + \ y_2) \geq 0$
   Ez pont annak a definiciója, hogy $z_1 + z_2 \in C$, $y_1 + y_2$ „tanuval”
2. $\lambda z_1 \in C$
   $\lambda z_1 = (\lambda y_1)(A|b), (\lambda y_1) \geq 0$
   Ez pont annak a definiciója, hogy $\lambda z_1 \in C$, (λy_1) „tanuval”
3. $(A|b)\ sorai \in C$
   i-dik sor tanuja $y_i = \begin{cases} y[j] = 1, ha\ j = i \\ y[j] = 0, ha\ j \neq i \end{cases}$

⇨ F-M minden lépése C-ben hagyja a mátrix sorait (természetesen az csupa 0 oszlopokat most nem hagyhatjuk el).

Ha $Ax \leq b$ nem megoldható, akkor (A|b)-n futtatva F-M algoritmust két lehetőség van:
Olyan sort kapunk, amelyre $0*x_n <=$ negatív => $(0..0| < 0)$!
Kaptunk két sort: $1 * x_i \leq \alpha, -1 * x_j \leq \beta, hogy\ \alpha < -\beta$, tehát
$(A|b)_i + (A|b)_j = (0\ ..0\ (1 + -1)|\alpha + \beta) = (0..0| < 0)$!

3. Farkas-lemma a lineáris egyenletrendszerek nemnegatív számokkal való megoldhatóságára („második alak"). A lineáris program célfüggvénye felülről korlátosságának feltételei.

Farkas lemma 2.

Tétel

Pontosan az egyiknek van megoldása:

(1) $Ax = b; x \geq 0$

(2) $yA \geq 0; yb < 0$

Bizonyítás

Kettőnek egyszerre nincs megoldása: $0 \leq (yA)x = y(Ax) = yb < 0$

(1) nem megoldható => (2) igen

(1)-et átírjuk $Ax \leq b; (-A)x \leq -b; (-E)x \leq 0$ egyenlőtlenségrendszerre:

$$\begin{pmatrix} A \\ -A \\ -E \end{pmatrix} x \leq \begin{pmatrix} b \\ -b \\ \bar{0} \end{pmatrix}$$

Farkas lemma 1 szerint ennek megfelelő másik egyenlőtlenségek:

$$y\begin{pmatrix} A \\ -A \\ -E \end{pmatrix} = \bar{0};\ \ y \geq 0;\ \ y\begin{pmatrix} b \\ -b \\ \bar{0} \end{pmatrix} < 0 \quad \text{ahol } y = (y_1|y_2|y_3)$$

$y_1A - y_2A - y_3E = \bar{0};\ y_1, y_2, y_3 \geq 0;\quad y_1b + y_2(-b) < 0$

$(y_1 - y_2)A = y_3;\ y_1, y_2, y_3 \geq 0;\quad (y_1 - y_2)b < 0$

Legyen: $\quad y' = y_1 - y_2$

$y'A = y_3;\ y_1, y_2, y_3 \geq 0;\quad y'b < 0$

$y'A \geq 0;\quad y'b < 0$

Tehát a Farkas lemma 1 értelmében, ha (1) nem megoldható, akkor létezik egy y'-t, ami megoldja (2)-t.

Célfüggvény korlátosság

3 kalitkás tétel

$Ax \leq b$ megoldható, ekkor az állítások ekvivalensek:

(1) Az $Ax \leq b$ megoldáshalmazon *cx* felülről korlátos. [És yb alulról korlátos]

(2) Nincs megoldása az $Az \leq 0; cz > 0$ rendszernek

(3) Van megoldása az $yA = c; y \geq 0$ rendszernek.

Bizonyítás

Elég belátni, hogy (1)→(2)→(3)→(1)

(1)→(2) indirekt

Legyen x_0 megoldása $Ax \leq b$-nek, és mégis létezik *z*, hogy: $Az \leq 0; cz > 0$.

Ekkor tetszőleges $x_0 + \lambda z;\ \lambda \geq 0$ is megoldás, mert $A(x_0 + \lambda z) = Ax_0 + \lambda(Az) \leq Ax_0 + 0 \leq b$. Továbbá $c(x_0 + \lambda z) = cx_0 + \lambda(cz)$ nem lehet korlátos felülről, mert $cz > 0$ és λ tetszőlegesen nagy lehet.

(2)→(3)

Farkas lemma 2 => Farkas lemma 2 transzponált:

$Bx = b; x \geq 0 \Leftrightarrow (Bx)^T = b^T; x^T \geq 0 \Leftrightarrow x^TB^T = b^T; x \geq 0$

$vB \geq 0; vb < 0 \Leftrightarrow (vB)^T \geq 0; (vb)^T < 0 \Leftrightarrow B^Tv^T \geq 0; v^Tb^T < 0$

x^T=y,B^T=A, ,b^T=c és v^T=-z behelyettesítéssel alkalmazható.

(3)→(1).

$cx = (yA)x = y(Ax) \leq yb$, minden *x, y*-ra *yb* felső korlát *cx* értékére.

*(3) miatt

**Eredeti feltevés miatt $Ax \leq b$

***Vegyük észre, hogy cx korlátossága egyben yb korlátosságát is jelenti.

4. A lineáris programozás dualitástétele (két alakban). A lineáris programozás alapfeladatának bonyolultsága (biz. nélkül).

Dualitás 1

Tétel

Ha $\max\{cx: Ax \leq b\}$ primál program megoldható és felülről korlátos, akkor

(1) $\min\{yb: yA = c; y \geq 0\}$ duális program is megoldható és alulról korlátos
(2) a primál programnak létezik maximuma, a duálisnak minimuma
(3) maximum = minimum

Bizonyítás

(1) bizonyítása:
A három kalitkás tétel lényegében bizonyítja az első állítást, azt kell belátni, hogy a korlátosság miatt létezik max/min és egyenlőek.
(2) és (3) bizonyítása:
Segéd állítás:
Tegyük fel, hogy $Ax \leq b$ megoldható és *t* tetszőleges.
Ha $Ax \leq b$-nek nincs olyan megoldása, amelyre $cx \geq t$ teljesülne, akkor $yA = c; y \geq 0$-nak van megoldása, amelyre $yb < t$ teljesül.
Bizonyítás:
Átírva:
$\begin{pmatrix} A \\ -c \end{pmatrix} x \leq \begin{pmatrix} b \\ -t \end{pmatrix}$
Farkas lemma 1 szerint ha ez nem megoldható, akkor az alábbi egyenlőség rendszer viszont igen:
Az abban szereplő (2) rendszer *y* vektor utolsó komponensét (ami a hozzáadott $(-c)x \leq -t$ egyenlőtlenség) válasszuk külön, jelölje λ. Ennek megfelelően Három kalitkás tétel:

$(y \quad \lambda)\begin{pmatrix} A \\ -c \end{pmatrix} = yA - \lambda c = 0 \rightarrow yA = \lambda c \qquad y \geq 0;\ \lambda \geq 0$

$(y \quad \lambda)\begin{pmatrix} b \\ -t \end{pmatrix} = yb - \lambda t < 0 \rightarrow yb < \lambda t$

Ha $\lambda = 0$ lenne, akkor $yA = 0; y \geq 0; yb < 0$ teljesülne, így a Farkas lemma alapján $Ax \leq b$ nem megoldható, ez ellentmond az állításnak.
Ezért $\lambda > 0$, így bevezethetjük $y' = \frac{1}{\lambda} y$. Erre $y'A = c$, $y' \geq 0$, $y'b < t$ teljesül, tehát *y'* teljesíti az állítást.
(2) bizonyítása primálra: Indirekt
Nem létezik maximum, de minden felülről korlátos halmaznak van szuprémuma (legkisebb felső korlátja), jelöljük: $t = \sup\{cx: Ax \leq b\}$. Mivel *t* szuprémum (=>felső korlát), ezért $Ax \leq b$-nek nincs $cx \geq t$-t teljesítő megoldása. Az előbbi állítás miatt $yA = c; y \geq 0$ rendszernek van $yb < t$-t teljesítő megoldása. Ekkor: $cx = (yA)x = y(Ax) \leq yb < t$. Itt *yb t*-nél kisebb felső korlát *cx*-re, ami ellentmondás, mert *t* a szuprémum.
(2) bizonyítása duálisra:
Egyszerűen átírjuk: $\max\{(-b)^T y^T: A^T y^T \leq c^T; (-A)^T y^T \leq (-c)^T; (-E)^T y^T \leq 0\}$ alakra.
(3) bizonyítása:
$\max\{cx: Ax \leq b\} \leq \min\{yb: yA = c; y \geq 0\}$ fennáll $cx \leq yb$ miatt.
Indirekt tegyük fel, hogy itt nem egyenlőség áll és legyen $t = \min\{yb: yA = c; y \geq 0\}$.
Indirekció miatt $cx \geq t$-nek nincs megoldása, a fenti tétel szerint ekkor a duálisnak van megoldása, amire $yb < t$, ami nyilvánvalóan nem lehet, mert *t* minimális.

Dualitás 2

Tétel

Ha $\max\{cx: Ax \leq b; x \geq 0\}$ primál program megoldható és felülről korlátos, akkor

(1) $\min\{yb: yA \geq c; y \geq 0\}$ duális program is megoldható és alulról korlátos
(2) a primál programnak létezik maximuma, a duálisnak minimuma
(3) maximum = minimum

LP bonyolultsága

- $Ax \leq b$-t kiegyenlítő *x* megoldások halmazán van-e *cx >= t*?
    - NP-beli: *x* tanú
    - coNP-beli: duális megoldása tanú
- szimplex módszer (1947) nem polinomiális, de gyors
- ellipszoid módszer (1979, Hacsijan) polinomiális, de lassú
- belső pontos módszerek (1984, Karmakar) polinomiális, de lassú

5. Egészértékű programozás: a feladat bonyolultsága, korlátozás és szétválasztás (Branch and Bound). Totálisan unimoduláris mátrix fogalma, példák ismert totálisan unimoduláris mátrixosztályokra. Egészértékű programozás totálisan unimoduláris együtthatómátrixszal (biz. nélkül)

Egészértékű programozás

Definiciók

- IP alapfeladat: $\max\{cx: Ax \leq b, x \text{ egész}\}$
- IP duálisa: $\min\{yb: yA = c, y \geq 0, y \text{ egész}\}$
- $\max\{IP\} \leq \max\{LP\} = \min\{DLP\} \leq \min\{DIP\}$

A feladat bonyolultsága

Eldöntési probléma

$Ax \leq b$-t kiegyenlítő *x* egész megoldások közül van-e, amire $cx \geq t$?

NP-beliség

NP-beli: x: $Ax \leq b, cx \geq t, x\ egész$ tanú

coNP-beliség

nincs dualitástétel: nem adódik a co-NP-beli.

NP-teljesség

Az NP-beliség triviális, tehét még azt kell belátni, hogy NP-nehéz, erre meg kell mutatni, hogy rajta keresztül megoldható egy NP-teljes probléma, például a k-Klikk (azaz Karp-redukálható az IP-re).
<u>k-Klikk redukálható IP-re:</u>
$k - Klikk := G(V,E)$: $\exists\ vc \subseteq V : (\bigcup_{(x,y)\in E: x\in vc\ \vee\ y\in vc}(x,y) = E)$
$k - Klikk - IP := \left\{\sum x_i \geq t\right\} x_i \begin{cases} = 1 => i - dik\ csúcs\ része\ klikknek \\ = 0 => i - dik\ csúcs\ nem\ rész\ klikknek \end{cases}$
$\forall(x_i)\ 0 \leq x_i \leq 1$
$\forall(v_i, v_j) \in \bar{E}(komplementár\ gráf\ élei)$: $x_i + x_j \leq 1$
(Ha nincs él köztük, akkor csak egyik csúcs lehet benne)

A Branch and Bound algoritmus

Program

$\max\{cx: Ax \leq b; f \leq x \leq g, x \text{ egész}\}$, *f* és *g* tetszőleges egész korlát vektorok.

Jelölések

- $\mathcal{L} = \{(\mathrm{IP}^{(i)}) = (f^{(i)}, g^{(i)}, w^{(i)})\}$ részfeladatok
- $w^{(i)}$: $IP^{(i)}$ maximumértéke ennél nagyobb nem lehet
- *z**: eddigi legjobb célfüggvényérték
- *x** eddigi legjobb megoldás ($cx^* = z^*$)

Az algoritmus lépései

0. $\mathcal{L} = \{(f, g, \infty)\}$ eredeti feladat, $z^* = -\infty$, x^* nem definiált.
1. Ha $\mathcal{L}$ üres, akkor STOP (megoldás *z**, *x** helyen). Egyébként vegyünk egy feladatot $\mathcal{L}$-ből (és töröljük onnan).
2. Ha $w^{(i)} \leq z^*$: IP$^{(i)}$ nem lehet megoldás, GOTO 1. (Bound lépés.)
3. IP$^{(i)}$ helyett LP$^{(i)}$ relaxált feladat megoldása. Ha nincs megoldás, akkor GOTO 1. Egyébként maximum $z^{(i)}$, maximumhely $x^{(i)}$.
4.
    a. Ha $z^{(i)} \leq z^*$, skip
    (A relaxált feladatnál nem lehet jobb, tehát már ismerünk az itt található lehetséges megoldásoknál jobbat.)
    b. Ha $z^{(i)} > z^*$, $x^{(i)}$ egész vektor. Ekkor z* és x* fölülírjuk ezeket az értékekkel.
    (Ez esetben találtunk egy jobb megoldást.)
    c. Ha $z^{(i)} > z^*$, $x^{(i)}$ nem egész vektor.
    (A relaxált feladat megoldása jobb, de nem egész, ezért ketté bontjuk és úgy próbálunk tovább vizsgálódni.)
    Ekkor válasszunk $x^{(i)}$-ben egy elágazási változót: x_j, és egy közbülső értéket:
    $f_j^{(i)} \leq t < g_j^{(i)}$.
    Definálunk két új korlátot f' és g', ahol $f_j' = t + 1$, másikban $g_j' = t$
    (Azaz x_j értékét egyik részfeladatban t-re alulról, másikban t+1-re felülről korlátozzuk)
    $\mathcal{L} += \{(f_j', g^{(i)}, z^{(i)}), (f^{(i)}, g_j', z^{(i)})\}$
    Ez a Branching lépés, az eredeti feladat helyett két új feladatot defináltunk, egyiket ahol x_j kisebb egyenlő t, és egyet ahol nagyobb egyenlő t+1-nél.

GOTO 1.

Az algoritmus hatékonysága

Az algoritmus véges sok lépésben leáll és megtalálja a feladat optimumát.
Bizonyítás:
<u>Véges:</u>
f és *g* miatt, mert csak véges sok egész *x* lehet úgy, hogy $f \leq x \leq g$. (csak f és g közti t értékek mentén tudunk elágazni.)
<u>Optimális:</u> indirekt
Legyen z_0 az optimum, de az eljárás csak $z^* < z_0$-t találta.
$\mathcal{L}$-ben mindig van feladat, aminek az optimuma z_0.
Ez a legelején (0. lépésnél) fönnáll.
Ezen optimális megoldás tartalmazó feladat esetén 4b vagy 4c lépéshez jut az algoritmus.
4b-nél meg is találja → ellentmondás.
4c esetén az egyik alfeladatban lesz benne az optimumhely (hisz egész értékű megoldás nincs t és t+1 között) és így optimumérték (z_0) is.
Így viszont $\mathcal{L}$ sosem ürülne ki, és az algoritmus sosem állna le, ellentmond az előző bekezdéssel.

Gyakorlati tapasztalatok az algoritmus használatára

- LIFO alapján válasszunk új feladatot $\mathcal{L}$-ből, mert a megoldás várhatóan mélyen van a fában, és LIFO-val tudunk a legmélyebbre hatolni.
- Elágazásnál a legkevésbé egész x_j-t válasszuk elágazási változónak (azaz amelyiknek a tört része 0,5-höz legközelebbi), közbülső értéknek pedig ennek egészrészét.

Totális unimoduláris (TU) mátrix

Definició

Minden négyzetes részmátrixának determinánsa 0, 1 vagy –1.
Szükséges (de nem elégséges), hogy a mátrix elemei is csak 0, 1 vagy –1 értékűek lehetnek.

TU tartó műveletek

Egy mátrix totális unimoduláris marad, ha

(1) egy sorát/oszlopát (–1)-gyel szorozzuk
(2) egységvektort hozzáveszünk sorként/oszlopként
(3) egyik sorát/oszlopát új sorként/oszlopként hozzávesszük
(4) transzponáljuk

TU tartás bizonyítása

Nyilván csak azoknak a négyzetes részmátrixoknak változhat a determinánsa, amelyikeket érint a változás.

(1) A determináns (–1)-szeres lesz (pl adott sor szerint kifejtve könnyű belátni).
(2) Ha a mátrixhoz egységvektort veszünk hozzá például oszlopként, és egy kiválasztott négyzetes részmátrixában ez az oszlop szerepel, akkor az új oszlop szerinti kifejtésből azonnal látszik, hogy a determináns megegyezik az eredeti mátrix egy négyzetes részmátrixának determinánsával vagy annak ellentettjével, így értéke 0, 1 vagy -1.
(3) Ha a régi és új sor/oszlop is szerepel a kiválasztott részmátrixnak, akkor a determináns 0. Ha csak az egyik (vagy a régi vagy az új), akkor előáll az eredeti mátrixból képzett részmátrix, determináns marad 0, 1 vagy –1.
(4) Determináns definíciójának következménye.

Egészértékű programozás totálisan unimoduláris együtthatómátrixszal

Legyen *A* totálisan unimoduláris mátrix, *b* egész koordinátájú vektor.
A $\max\{cx: Ax \leq b\}$ (LP) feladat megoldható és korlátos.
Ekkor a $\max\{cx: Ax \leq b, x \text{ egész}\}$ (IP) feladat is megoldható, és maximuma megegyezik (LP) feladat maximumával, azaz az (IP) feladatnak létezik egész maximum helye.
$\max\{IP\} = \max\{LP\} = \min\{DLP\} \leq \min\{DIP\}$
$\max\{IP\} = \max\{LP\} = \min\{DLP\} = \min\{DIP\}$, akkor ha c is egész vektor.
(A DIP-ben a „c” a b megfelelője.)

Illeszkedési mátrix

Legyen az n-pontú G irányított gráfnak e éle és definiáljuk az n*e méretű B(G)=(bij) mátrix elemeit úgy, hogy:

$$b_{ij} = \begin{cases} 0 & ha\ a\ j-edik\ él\ nem\ illeszkedik\ az\ i-edik\ ponthoz \\ 1 & ha\ a\ j-edik\ élnek\ az\ i-edik\ pont\ a\ kezdőpontja \\ -1 & ha\ a\ j-edik\ élnek\ az\ i-edik\ pont\ a\ végpontja \end{cases}$$

Legyen $b_{ij} = 1$ akkor is, ha a j-edik él az i-edik ponthoz illeszkedő hurokél.
Irányítatlan esetben is ez a definíció, csak ott a j-edik él mindkét végpontjának megfelelő mátrixelem 1.

Irányított gráf illeszkedési mátrixa TU

Teljes indukció: *M* egy $k \times k$ méretű részmátrix.

- Ha $k = 1$, akkor nyilvánvaló, mert minden elem 0, -1 vagy 1.
- Ha $k \geq 2$, és

- *M*-nek van olyan oszlopa, amelyben legfeljebb egy nemnulla elem van, akkor fejtsük ki detM-et e szerint az oszlop szerint, ekkor az indukciós feltétel szerint készen vagyunk. (Ha létezik hurok él a gráfban.)
- Egyébként minden oszlopban egy +1 és egy –1 elem van, ekkor *M* sorainak összege nullvektor, a determináns 0.
  (Hiszen minden oszlop egy él, amihez egy kezdő és egy végpont tartozik pontosan, kivéve ha hurok él.)

Páros gráf illeszkedési mátrixa TU

Felhasználjuk az irányított gráf tételét:
$G = (A, B; E)$ páros gráf éleit irányítsuk úgy, hogy minden él *A*-ból *B*-be mutasson, ekkor B(G') TU.
A *B*-hez tartozó sorokat negáljuk, de ez nem változtat TU tulajdonságon viszont visszakapjuk az eredeti irányítatlan gráf illeszkedési mátrixát.

6. A lineáris és egészértékű programozás alkalmazása páros gráfokra, Egerváry tétele.

Alkalmazás páros gráfokra

Legyen n csúcsú, m élű irányítatlna G gráf illeszkedési mátrixa *B* és tenkintsük *Bx <= (1,1,…..,1)^T, x>=0* rendszer egész megoldásait. Látható, hogy minden ilyen megoldás 0-1 értékű. Az is könnyen ellenőrizhető, hogy egy 0 és 1 komponensekből álló *x* vektor pontosan akkor megoldás, ha az 1 komponenseknek megfelelő élek független élhalmazt alkotnak G-ben. Ezért ha most *w* tetszőleges *m* dimenziós vektor, akkor a *max{wx: Bx<=(1,1,….,1^T), >=0, x egész}* (IP) feladat nem mást jelent, mint hogy G-ben maximális súlyú független élhalmazt (párosítást keresünk), ahol az élek súlyait *w* tartalmazza.

Tegyük fel most, hogy G páros gráf. Ekkor az tétel (Páros gráf illeszkedési mátrixa totálisan unimoduláris) szerint B totálisan unimoduláris. A fenti (IP) feladatot leíró mátrixot (az x >= 0 feltétel miatt) úgy kapjuk, hogy B-t kiegészítjük az m * m-es egységmátrix ellentettjével, ez azonban. lemma (TU megmaradás) szerint a totális unimodularitáson nem változtat. Így az (IP) feladat maximuma megegyezik a megfelelő (LP) feladat maximumával. Erre viszont már alkalmazhatjuk a dualitástételt:

$$\max\{wx: Bx \leq (1,\dots,1)^T, x \geq 0\} = \min\{y(1,\dots,1)^T : yB \geq w, y \geq 0\}$$

Mivel a duális feladat megoldásának koordinátái a *B* sorainak felelnek meg, ezért egy *y* megoldás minden *v* csúcshoz egy *c(v)* címkét rendel. A címkékre nézve *yB>=w* azt jelenti, hogy ha az $e \in E$ él két végpontja *x* és *y,* akkor $c(x) + c(y) \geq w(e)$ teljesül. Az alábbi alakban is felírható.

Egerváry Jenő tétel

Legyen $G = (A, B; E)$ páros gráf és minden $e \in E$ élhez legyen adott egy *w(e)* súly. Ekkor a maximális összsúlyú párosítás összsúlya:

$$min \sum_{v \in A \cup B} c(v)$$

ahol a fenti minimum az összes olyan, a csúcsokon értelemzett, nemnegatív értékű c függvényen értendő, amelyre $c(x) + c(y) \geq w(e)$ telejseül minden *e = {x,y}* élre.

Tömören

Jelölések: *x* indikátor, hogy egy él benne van-e a párosításban. *w* tetszőleges (él)súlyfüggvény, *B* illeszkedési mátrix (páros gráf lévén TU).

Alkalmazás páros gráfokra

$\max\{wx: Bx \leq (1,1,\dots,1)^T, x \geq 0\}$.
Az $x \geq 0$ feltétel miatt *B*-t kiegészítjük az $m \times m$-es egységmátrix ellentettjével, de ez nem változtat TU tulajdonságon.
A $Bx \leq (1,1,\dots,1)^T$ azt jelenti, hogy az egy csúcsból kiinduló élek közül legfeljebb egyet választunk ki → párosítás.

Duális (Egerváry Jenő tétele)

$\max\{wx: Bx \leq (1,1,\dots,1)^T, x \geq 0\} = \min\{y(1,1,\dots,1)^T: yB \geq w; y \geq 0\}$.
Ekkor *y* megoldás minden *v* csúcshoz egy címkét rendel.
A $yB \geq w$ azt jelenti, hogy $c(a) + c(b) \geq w(e)$ minden $e = \{a, b\} \in E$ élre.
(Egerváry tétele lényegében a max=min dualitását mondja ki csak LP nélkül megfogalmazva.)

7. A lineáris és egészértékű programozás alkalmazása hálózati folyamproblémákra: a maximális folyam, a minimális költségű folyam és a többtermékes folyam feladatai, ezek hatékony megoldhatósága a tört-, illetve egészértékű esetben

Jelölések

$G = (V, E)$ irányított gráf

- $s, t \in V$ két kitüntetett csúcs: forrás ás nyelő
- $c: E \to \mathbb{R}_+$ nemnegatív *kapacitásfüggvény*.
- $x: E \to \mathbb{R}_+$ tetszőleges függvény.
- $in(v)$: *v*-be belépő élek összege *x* szerint
- $out(v)$: *v*-ből kilépő élek összege *x* szerint
- *x folyam*, ha $\forall\, v \in V \backslash \{s, t\} :\ \ in(v) = out(v)$
- *x* megengedett, ha $\forall e \in E :\ x(e) \leq c(e)$
- A folyam értéke $in(s) - out(s) = in(t) - out(t)$
- B a G illeszkedési mátrixa

Maximális folyam, mint LP

Program

$B' \coloneqq B \setminus \{s, t\ sorai\} \sim hiszen\ ezekre\ nem\ vonatkozik\ a\ Kirchoff$ törvény
$x[i] \coloneqq x(e_i) \sim$ tehát az $i.$ változó értéke az $i.$ él értéke
$B'x = \ v[i] \coloneqq in(e_i) - out(e_i) \sim azaz\ a\ B'x\ vektor\ megadja\ a\ i.$ élen átfolyó $folyam$ értéket
$\max\{out(s) - in(s)\} = \max\{in(t) - out(t)\} \sim folyamértékét\ maximalizáljuk$

$$\begin{matrix} B'x \leq 0 \\ -B'x \leq 0 \end{matrix} \sim Kirchoff\ törvénye, ami\ kifolyik\ az\ is\ be\ is\ folyt$$

$$\begin{matrix} Ex \leq c(e) \\ -Ex \leq 0 \end{matrix} \sim adott\ élen\ folyó\ érték\ min\ 0\ és\ max\ a\ kapacitása$$

TU tulajdonság

A B mátrixról ismert, hogy TU továbbá TU-tartó műveletekkel előállítható a teljes IP mátrix:

1. Felvesszük B minden sorát még egyszer
2. Az újonnan felvett sorokat -1-vel szorozzuk
3. Felvesszük az egység mátrix sorait kétszer
4. Szorozzuk egyik példányát minden egységvektornak -1-vel

Az IP feladat mátrixa tehát TU, a hasonlítási vektor 0 és c(e) értékeket tatalmaz, tehát ha a kapacités egész, akkor van egész értékű maximális folyam is.
(Ha egyáltalán létezik folyam rajta és korlátos.)

Minimális költségű folyam keresésére

Feladat

$k: E \to\ \mathbb{R}^+ \sim e\ él\ bevételének\ a\ költsége$
Cél: $\min_{e \in E} k(e)\,, in(t) - out(t) \geq M\ (tetszőleges\ érték)$
Tehát a legalább M költségű folyamok közül keressük a legolcsóbbat.

Program

$\min\{kx: B^*x \leq 0, in(t) - out(t) \leq -M\}$
B* legyen a folyam és kapacitás feltételeket leíró TU mátrix amit már ismertettünk.

TU tulajdonság

Az $in(t) - out(t) \geq M$ még egy sor, de ettől még TU marad, hisz lényegében az eredeti B illeszkedési mátrix t-nek megfelelő sorát kell csak visszavenni hozzá.

Alkalmazás többtermékes folyamra

Feladat

(s,t) helyett k db (s_i,t_i) forrás és nyelő páros
x(i,j) := i-dik élen, a j-dik termélből mennyi folyik át ~ k*e változónk van
(Tehát k termékünk van, az i-dik termék csak s_i-ben termelődik és t_i-ben nyelődik el, a kapacitás viszont az összes termékre egységesen vonatkozik.)

Program

A Kirchoff feltételek ugyanúgy kifejezhetőek (i-dik termék esetén csak az (s_i,t_i) sorokat hagyjuk el).
Arra kell figyelni, hogy most k*e változónk van míg az illeszkedési mátrix jobb dimenziója e, azaz ahelyett hogy egymás allá raknák az adott mátrixokat egymás mellé tudjuk ragasztani és a sor elhagyása helyett az i-dik termék mátrixában az (s_i,t_i) sorait nullvektorral helyettesítjük.
A kapacitás feltételek viszont újjelegűek:
$0 \leq \forall e \in E, i \in [1,k] : \sum x(e,i) \leq c(e) \sim (e\ helyett\ e\ sorszáma\ kéne\ formálisan)$

TU tulajdonság

Mivel mindig más sort nullázunk ki, így már nem beszélhetünk arról, hogy már létező olszopot veszünk hozzá plusszba és így a TU tulajdonság sem garantált.

8. Polinomiális időben megoldható feladat fogalma, példák. Az NP, co-NP, NP-nehéz és NP-teljes problémaosztályok definíciója, viszonyaik, példák problémákra valamennyi osztályból. NP-nehéz feladatok polinomiális speciális esetei: algoritmus a maximális független ponthalmaz problémára és az élszínezési problémára páros gráfokon. Additív hibával közelítő algoritmusok speciális pont-, illetve élszínezési problémákra.

Eldöntési probléma osztályok

Definiciók

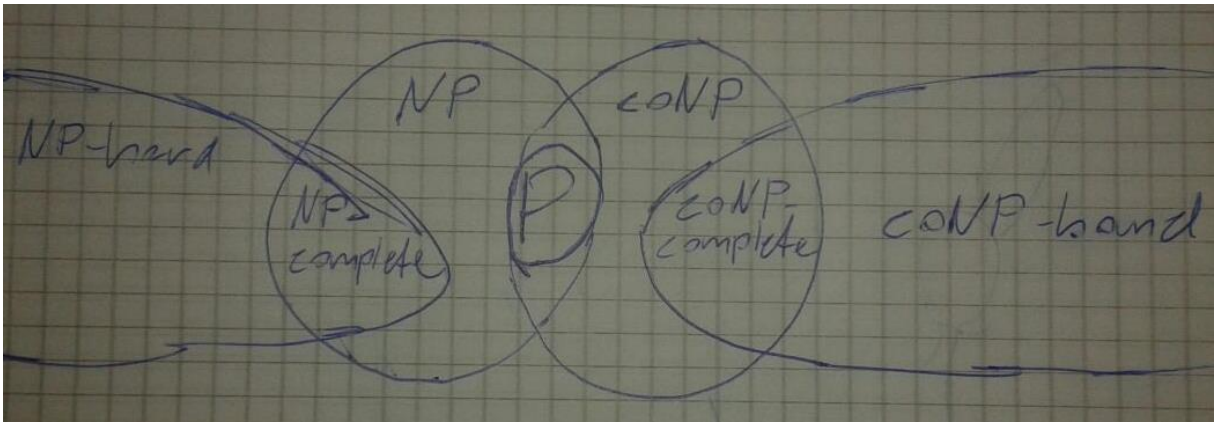
Csak eldöntési problémákra van definálva (megadható pl. az IGEN-t adó bemenetek halmazával és az összes bemenetet leírni képes abécé-vel.)
A probléma osztályok:

- P: polinomiális időben megoldható
- NP: ha a válasz IGEN, akkor létezik egy tanuja, ami segítségével polinomiálisan belátható, hogy tényleg ez a válasz.
- co-NP: ha a feladat negáltja NP (azaz ua, mint NP, csak a NEM válaszra kell tanu létezzen).
- NP-nehéz: minden NP-beli probléma Karp-redukálható rá
  (azaz ha adott egy NP-beli probléma, annak minden bemenetét át tudjuk polinomiális időben alakítani egy saját bemenetünkre, hogy az adott válasz nem változik)
  (Ezzel azt látjuk be, hogy az adott feladat nem lehet a sajátunknál lényegesen nehezebb, ha ezek közül bármelyiket meg tudnánk oldani P-ben, akkor az összes többi is P-beli lenne.)
- NP-teljes: NP-beli és NP-nehéz

Viszonyaik

Nyitott kérdések:

- $P = NP$ (sejtés: nem, sok NP-beli problémára nem ismert P-beli megoldás)
- $P = NP \cap coNP$ (sejtés: igen, legtöbb problémára ismert P-beli megoldás)



Híres eldöntési problémák

feladat	probléma osztály	algoritmus
összefüggőség	P	mélységi keresés (DFS), szélességi keresés (BFS)
TP páros gráfban	P	javító utas algoritmus
2-Szín	P	BFS
3-Szín	NP-teljes	-
Síkbarajzolhatóság	P	Kuratowski-tétel
k-Klikk	NP-teljes	-
k-Ftln	NP-teljes	-
k-Ftln páros gráfon	P	magyar módszer
Hamilton-kör	NP-teljes	-
utazóügynök probléma	NP-teljes	-
k-Hosszú kör	NP-teljes	
leghosszabb irányított út keresése	NP-nehéz (nem eldöntési)	-
leghosszabb irányított út keresése DAG-ban	P	szélességi keresés (BFS)
Gráf izomorfizmus	NP	*(Nem ismert, hogy P vagy NP-teljes)*
Megállási probléma	NP-nehéz (nem NP!)	Véges időben eldönthetetlen

Független és lefogó halmazok

Jelölés

- $\nu(G)$ a legnagyobb független élhalmaz elemszáma
- $\tau(G)$ a legkisebb lefogó ponthalmaz elemszáma
- $\rho(G)$ a legkisebb lefogó élhalmaz elemszáma
- $\alpha(G)$ a legnagyobb független ponthalmaz elemszáma

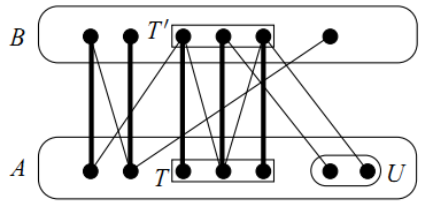
Viszony

- $\nu(G) \leq \tau(G)$
- $\alpha(G) \leq \rho(G)$
- $\tau(G) + \alpha(G) = |V(G)|, ha\ G\ hurokmentes\ [Gallai\ 1]$
- $\nu(G) + \rho(G) = |V(G)|, ha\ G - ben\ nincs\ izolált\ csúcs\ [Gallai\ 2]$
- $\nu(G) = \tau(G), páros\ gráfban\ (hurokél\ emiatt\ nem\ lehet)[König]$
- $\rho(G) = \alpha(G), páros\ izolált\ pont\ mentes\ gráfban\ [König]$

NP-nehéz feladatok polinomiális speciális esetei

Maximális független ponthalmaz páros gráfokon

Futtassuk a magyar módszert, ha már nincs javító út, akkor az alábbi helyzet alakul ki:



A, B	Páros gráf csúcsosztályai
M	A jelenlegi párosítás
U	A\M (A csúcsok közül az M által fedetlenek)
T’	U-ból alternáló úton elérhető B csúcsok
T	T’ párosításai

(B \ T’) u (T u U) független csúcshalmaz, hisz

- (T u U)-ból induló élek T’-be kell, hogy menjenek (különben lenne javító út).

T’ u (A \ (T u U)) lefogó csúcshalmaz, hisz:

- (T u U)-ból induló élek T’-be kell, hogy menjenek (különben lenne javító út).
- A csúcshalmaz többi csúcsa pedig a lefogó csúcshalmaz része.

|T’ u (A \ (T u U))| = |M|, hisz:

Minden párosításbeli élnek pontosan egy csúcsát tartalmazza.

T’ u (A \ (T u U))-ról tudjuk, hogy minimális lefogó, hisz egy párosításnak minden éléhez legalább egy csúcsot tartalmaznia kell (azaz >= |M|).

Gallai tétel értelmében viszont |max független csúcshalmaz| + |min lefogó csúcshalmaz| = |V(G)|, de [(B \ T’) u (T u U)] u [T’ u (A \ (T u U))] = A u B, azaz a talált független ponthalmaz teljesíti ezt a feltételt, ezért maximális.

Élszínezés páros gráfokon

König tétele: χe(G) = Δ(G).
Ennek bizonyítása egyben algoritmus arra, hogy Δ(G) hogy lehet kiszínezni páros gráfot P-ben:
G(A,B,E) gráf éleit sorban színezzük, mindig helyesen, ezalatt esetleg átszínezve a már színezetteket (de színtelenné nem alakítjuk).
e_1: teljesen mindegy mire színezzük
$e_i = (u \in A, v \in B)$: tfh. e_{i-1}-ig már helyesen színezett
Definició szerint mind u és v csúcsra is max Δ(G) él illeszkedhet.
De maximum csak Δ(G)-1 szín lehet használatban mindkét csúcsban jelenleg, hisz legalább az e_i él még színtelen, azaz mindkét csúcsban még van szabad szín(ek).
Ha ezek között van egyezés, azt a színt megkaphatja e_i.
Ellenkező esetben legyen u egyik szabad színe piros, v-é meg kék.
Konstruáljuk G azon rész gráfát C-t, amiben csak azon élek és azok csúcsai vannak amellyek kék vagy piros színűek. I C minden csúcs foka maximum 2 (egy piros és egy kék él), azaz diszjunk utak és körökre bontható.
u és v fokai viszont maximum egy (különben lett volna egyező szín), azaz egy-egy út végpontjai lehetnek csak.
Legyen az u-t tartalmazó út U, ezen nem lehet rajta v, hisz csak másik végpontja lehet, de az adott út ekkor páratlan hosszú lenne (egyik színosztályból a másikba vezet) és váltakozó színú (hisz a színezes helyes), azaz ha az első él kék (u-ban piros volt szabadon), akkor az utolsó is, azaz v-ben is piros szín szabad lenne.
Tehát ekkor az U mentén felcserélhetjük a színeket, ezzel nem elrontva a színezést, de u-ban piros helyett a kék szín lesz szabad így, amivel e_i színezhető.

Additív hiba

Definició

$I\ input \rightarrow\ x_I\ megoldáshalmaz$
$c\colon x_I \rightarrow\ \mathbb{R}\ megoldást\ értékelő\ függvény$
$C\colon konstans\ additív\ hiba$
$x'\ :\ közelítő\ algoritmus\ által\ talált\ legjobb\ megoldás$
C-Additív hibától eltekintve helyes egy megoldás, ha
Maximalizálásnál:
$\forall I\colon c(x') \geq \max_{x_i \in x_I} c(x_i) - \mathrm{c}$
Minimalizálásnál:
$\forall I\colon c(x') \leq \min_{x_i \in x_I} c(x_i) + \mathrm{c}$

C-Additív közelítő algoritmus

Olyan A algoritmus, amely képes C-Additív hibától eltekintve helyes megoldást adni POLINOMIÁLIS lépésszámban.
(A közelítő megoldás értéke nem elég, a konkrét x' megoldás kell.)

Élszínezés 1-additív hibával

Ismert Vizing tétele szerint, hogy $\Delta(G) \leq \Delta\chi e(G) \leq \Delta(G) + 1$ (egyszerű gráfon).
Létezik algoritmus, ami $\Delta(G) + 1$ színnel polinomiális időben kiszínezi egy egyszerű gráf éleit.

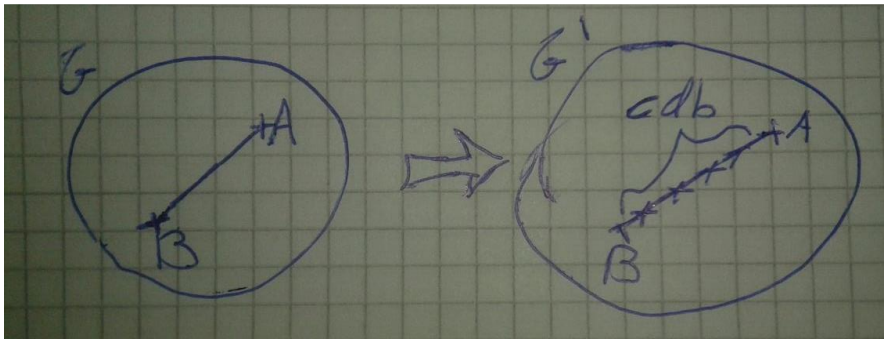
Csúcsszínezés 1-additív hibával síkba rajzolható gráfokon

Négyszíntétel értelmében 4 színnel biztos színezhetőek és erre létezik is polinomiális algoritmus.
Az additív hiba 3-ről 1-re csökkenthető, ha előtte egy DPS futtatásával meggyőződünk arra, hogy páros-e.
Ha páros, akkor a DPS mentén egy színezést is találtunk és így nincs hiba.
Ha nem páros, akkor 4 színnel színezzük, de eze setben a kromatikus szám legalább 3, azaz biztosak lehetünk benne, hogy max 1 a hiba.

9. A Hamilton-kör probléma visszavezetése a leghosszabb kör probléma additív közelítésére. k-approximációs algoritmus fogalma, példák: két-két algoritmus a minimális lefogó ponthal-maz keresésére és a maximális páros részgráf keresésére.

Hamilton-kör visszavezetése leghosszabb kör probléma additív közelítésére

A Hamilton-kör eldöntési problémát visszavezetjük a leghosszabb kör c-additív közelítésére.
G -> G': az eredeti gráf minden élén felveszünk c db új csúcsot:



Vegyük észre, hogy G' és G körei egymásnak bijektíven megfeleltethetőek.
Ráadásul G' egy köre az eredeti kör hosszának c+1-szerese.
Ennek következménye, hogy G köreinek a hosszai „c+1"-es léptékekben nőnek, ami c additív hiba megengedése mellett is azt jelenti, hogy a mindig a leghosszabbat kell megtalálnunk.

Tehát ha találtunk egy k*(c+1) hosszú kört G-ben az megfelel egy Hamilton-körnek G-ben.
Ha létezik Hamilton kör G-ben, akkor G'-ben létezik k*(c+1) hosszú kör, a nála rövidebb kör már csak (k-1)*(c+1) hosszú lehet csak, ami nem fér bele a c-additív hibába.

Tehát ha az additív problémát meg tudjuk oldani polinomiális időben, akkor a Hamiltön-kör problémát is meg tudnánk oldani polinomiális időben. (Ehhez még fontos azt is konstatálni, hogy az átalakítás maga megy polinomiáls időben.)

Multiplikatív hiba

Definició

$I\ input \rightarrow\ x_I\ megoldáshalmaz$
$c\colon x_I \rightarrow\ \mathbb{R}\ megoldást\ értékelő\ függvény$
$k\colon \geq 1\ konstans\ multiplikatív\ hiba$
$x'\ :\ közelítő\ algoritmus\ által\ talált\ legjobb\ megoldás$
k-multiplikatív hibától eltekintve helyes egy megoldás, ha
Maximalizálásnál:

$$\forall I\colon c(x') \geq \frac{1}{k} * \max_{x_i \in x_I} c(x_i)$$

Minimalizálásnál:

$$\forall I\colon c(x') \leq k * \min_{x_i \in x_I} c(x_i)$$

k-approximációs algoritmus

A egy feladat k-approximációs algoritmusa, ha polinomiális időben képes egy feladatot k-multiplikatív hibától eltekintve helyesen megoldani.

Maximális páros részgráf 2-approximációja

Keressük *n* pontú gráf olyan kettéosztását, ahol a két pontosztály között a legtöbb él halad.
Kiindulunk egy tetszőleges kettéosztásból, és ha egy *p* pont áthelyezése növeli a vágás élszámát, akkor áthelyezzük.
Nem biztos, hogy minden csúcsot csak egyszer kell áthelyeznünk (egy későbbi áthelyezés miatt lehet mégis vissza akarnánk tenni), de azt tudhatjuk, hogy minden áthelyezésnél legalább egy éllel több fog menni a két színosztály között, azaz O(E) lépés után biztos véget ér.
Jelölje inner(v) = a v csúcsból saját osztályán belüli csúcsba vezető élek számát,
outer(v) = a v csúcsból a másik osztályba vezető élek számát.
Az algoritmus akkor áll le, ha minden csúcsra igaz, hogy: outer(v) >= inner(v).
Ekkor viszont
Σouter(v) >= Σinner(v) és
Σouter(v) + Σinner(v) = |V|,
tehát
Σouter(v) >= |V|/2 >= |OPT|/2.

Maximális páros részgráf online 2-approximációja

Online algoritmus, létrehozunk két üresosztály és sorban beosszunk az összes csúcsot valamelyikbe.
Mohón osztunk be a két osztály közül abba helyezzük, amelyikkel végül több keresztélet kapunk.
Itt nem igaz, hogy outer(v) >= inner(v) viszont Σouter(v) >= Σinner(v) továbbra is igaz, hisz minden elhelyezésnél úgy döntünk, hogy a Σouter(v) legyen többségben.

Minimális lefogó ponthalmaz 2-approximációja

Keressünk egy optimális párosítást, ezen csúcsait választjuk.
Lefogó:
Ha nem lenne, akkor létezne egy él amely egyik csúcsában sem csatlakozik bármelyik párosított élhez, azaz hozzá tudnánk venni a párosításhoz, de ez ellentmondás, hisz maximális a párosításunk.
2-approximáció:
τ jelölje a minimális lefogó csúcshalmaz méretét és ν a maximális független élek számát.
A mi párosításunk maximális, azaz és ν méretű, azaz a választott lefogó csúcshalmazunk 2ν elemű.
Ismert, hogy $\nu \leq \tau$, ebből következik, hogy $sol = 2 * \nu \leq 2 * \tau = 2 * OPT$, ami a 2-approximáció definiciója.

Minimális lefogó ponthalmaz 2-approximációja mohó algoritmussal

Vegyük észre, hogy a lefogó csúcshalmaz bizonyításakor csak annyit használtunk, hogy a párosítás nem bővíthető tovább és a 2-approximációnál felhasznált egyenlőtlenség pedig minden párosítás élszáma és lefogó csúcshalmaz csúcsszáma között fennáll nem csak a maximum és minimum között.
Tehát nem kell optimális párosítás, bőven elég nem bővíthető, ezt pedig egy mohó algoritmussal is meg tudjuk keresni.

10. A minimális lefogó ponthalmaz probléma visszavezetése a halmazfedési feladatra, a halmazfedési feladat közelítése, éles példa. Közelítő algoritmus a Steiner-fa problémára, éles példa.

Súlyozott halmazfedés

Feladat

$U \sim alaphalmaz\ (bármi\ lehet\ igazából)$
$R = \{S_i \mid S_i \subseteq U\}, |R| = t \sim U\ valahány\ nem\ feltétlenül\ diszjunkt\ részhalmazai$
$\bigcup_{S_i \in R} S_i = R \sim azaz\ olyan\ részhalmaz\ rendszert\ választunk, amivel\ fedhető\ az\ alaphalmaz$
$c : R \rightarrow \mathbb{Q}^+ \sim egy\ adott\ részhalmaz\ választásának\ költsége$
$find\ I \subseteq R : \min_{i \in I} c(S_i)\ \&\&\ \bigcup_{i \in I} S_i = R$
$\sim tehát\ keressük\ azon\ részhalmazokat, amik\ együtt\ már\ lefedi\ az\ egész\ alaphalmazt$
$és\ összköltségben\ minimálisak$

Minimális lefogó ponthalmaz probléma visszavezetése a halmazfedési feladatra

$U = E$
$R = \{\{(x,y) | x = v_i \vee y = v_i\} | v_i \in V\} \sim tehát\ a\ részhalmazaink\ egy -$
$egy\ csúcshoz\ tartoznak\ és\ az\ adott\ csúcsból\ kiinduló\ éleket\ tartalmazzák$
$c : R \rightarrow 1 \sim minden\ csúcs\ bevétele\ egységnyi$ árú
R egy-egy eleme pont egy-egy csúcsnak felel meg, hogy melyikeket válasszunk be a lefogó csúcshalmazba és azon élek halmazából áll, amiket az adott csúcs lefog.
Ebből következik az is, hogy a halmaz fedés NP-nehéz.

A halmazfedés közelítése

Legyen C ⊆ U a már eddig fedett értékei az U alaphalmaznak.
Ekkor mohó algoritmust alkalmazunk a legkisebb $\frac{c(S_i)}{|S_i \backslash C|}$ érték szerint.
(A $c(S_i)$ a halmaz költsége, $|S_i \backslash C|$ a részhalmaz által fedhető elemek száma, azaz a lefedett elem egységköltségét próbáljuk minimalizálni mohó módon. A \C jelentősége, hogy amit már egy másik halmazzal lefedtünk, azt nem akarjuk még egyszer lefedni, csak az újonnan lefedett elemek száma érdekel minket.)

Közelítésének faktora

$A\ mohó\ fedés\ költsége \leq \left(\frac{1}{1} + \frac{1}{2} + \cdots \frac{1}{r}\right) * OPT = H_r * OPT \leq (\ln(r) + 1) * \mathrm{OPT}$
$r = \max_{S_i \in R} |S_i| \sim a\ legnagyobb\ részhalmaz\ mérete$
Bizonyítás:
$p_f : U \rightarrow \mathbb{Q}^+ \coloneqq p(e) = \frac{c(S_i)}{|S_i - C|}, ahol\ S_i\ e - t\ legelőször\ fedő\ részhalmaz\ az\ f\ fedés\ szerint$
Az előbb definált újonnan behozott elem egységárt az elemekre is definálják.
Ekkor egy f fedés öszköltsége az elemek f fedés szerinti költségüknek az összege.
Vizsgáljuk meg egy $S \in R_{OPT}$ halmazát egy optimális fedésnek.
Indexeljük S elemeit amilyen sorrendben a f mohó algoritmus választotta őket és vizsgáljunk meg egy tetszőleges i-dik elemet (i <= r).

Ekkor $p_f(s_i \in S) = \frac{c(S')}{|S' \setminus C|} \leq \frac{c(S)}{i}$, hiszen mikor s_i-t fedni készül a mohó algoritmus, akkor S még i új elemet tudna lefedni, tehát a költsége $\frac{c(S)}{i}$, ha S' költsége ennél nagyobb lenne, akkor S' helyett S-t választotta volna ezen a ponton.

Az i és S választása tetszőleges, tehát egy S-beli elemeket a mohó algoritmus legfeljebb: $\sum_{i=1}^{r} \frac{c(S)}{i} = c(s) * \sum_{i=1}^{r} \frac{1}{i} = c(s) * H_r$ költséggel fedi.

Azaz az egész halmazt pedig:

$$\sum_{S \in R_{OPT}} c(S) * H_r = H_r * OPT$$

Ez pont annak a deficiója, hogy az közelítési faktor H_r.

Közelítés éles példája

$U = [1..n]\ (csak\ egész\ számok)$

$R = \{\{i\} \mid i \in [1..n]\} \cup U \sim tehát\ a\ részhalmazok\ egy -$

$egy\ számot\ tartalmazó\ halmazok\ és\ az\ egész\ alaphalmaz\ együtt$

$$c := \begin{cases} \frac{1}{i}, ha\ \{i\} \sim \\ 1 + \varepsilon \end{cases} \begin{matrix} Az\ egy\ elemű\ halmazók\ költsége\ a\ benne\ lévő\ elem, \\ a\ teljes\ halmazé\ pedig\ egy\ előre\ nem\ definált\ 1 + \varepsilon \end{matrix}$$

ε értékét úgy választjuk meg, hogy elég kicsi legyen ahhoz, hogy az optimális megoldás a teljes U-t válassza, de elég nagy legyen ahhoz, hogy a mohó algoritmus ne válassza a teljes halmazt.

Választott részhalmaz	$\{n\}$	$\{n-1\}$	$\{i\}$	$\{1\}$
Költsége*	$\frac{1}{n}$	$\frac{1}{n-1}$	$\frac{1}{i}$	1
u költsége	$\frac{1+\varepsilon}{n}$	$\frac{1+\varepsilon}{n-1}$	$\frac{1+\varepsilon}{i}$	$1+\varepsilon$

A mohó algoritmus költsége épp H_r lesz, míg az optimum a $1+\varepsilon$ lenne.

$\lim_{\varepsilon \to 0} \frac{H_r}{1+\varepsilon} = H_r$ => minden H_r-nél nagyobb számra találunk olyan kicsit ε-t, amire már elrontjunk a példát, tehát ennél kisebb faktorú közelítésre ez a mohó algoritmus nem képes.

*Vegyük észre, hogy itt igazából 1/i osztva 1 (mert egy új elemet hoz be) szerepel!

Steiner-fa

Feladat

$G = (V, E)\ egyszerű, összefüggő$

$c: E \to \mathbb{R}_+ \sim élköltség\ függvény$

$S, T \subseteq V, S \cup T = V, S \cap T = \emptyset \sim adott\ V - nek\ egy\ diszjunk\ felosztása$

$T \sim terminálok, S \sim Steiner\ pontok$

$Steiner - fa := F \subseteq V\ fa: T \subseteq F$

$\sim olyan$ részfa, ami a terminálokat feszíti, a Steiner pontokból meg tetszőleges számban tartalmazhat

$find\ F: \min_{F\ Steiner\ fa} \{c(F)\} \sim tehát\ a\ minimális$ összköltségű $Steiner - fát\ keress$ük

(Ilyen természetesen létezik, hisz összefüggő gráfnak létezik feszítőfája, ami Steiner-fa is.)

Bonyolultság

$T = V$ esetén közismerten P-beli, mohó algoritmussal megoldható.

$|S| = konstans$ esetén a lehetséges feszítőfákat besorolhatjuk aszerint, hogy a Steiner pontok közül melyeket használja így kapunk $2^{|S|}$ részfeladatot (ennyir részhalmaza létezik S-nek), ezeken már elég egy egyszerű minimális

feszítőfát keresni majd a részfeladatok optimumai közül kiválasztani a legjobbat. (Ez megy polinomiális időben, hisz $2^{|S|}$ konstans).
$|S| = konstans * \log(n) = \log(n^c)$ esetén a részfeladatok száma $2^{|S|} = 2^{\log(n^c)} = n^c$ így összeségében még mindig polinomiáls marad a lépésszám.
Általánosságban viszont NP-teljes problémáról van szó.

Metrikus gráf

$G\ gráf\ metrikus, ha\ teljes\ és\ az\ élsúlyozás\ teljesíti\ a\ háromszög - egyenlőtlenséget.$
$\forall(x,y),(y,z),(x,z) \in E: c\big((x,y)\big) + c\big((y,z)\big) \geq c((x,z))$

2-approximáció metrikus gráfon

Algoritmus:
T-ben keresünk minimális feszítőfát (tudunk, mert G teljes).
Bizonyítás:
1. T_0* := optimális Steiner-fa
2. T_1* := T_0* minden élet megduplázzuk ~ c(T_1*) = 2*c(T_0*)
3. Keres T_1* Euler-sétáját
4. T_2* := T_1* részleges Euler bejárása, csak a T-beli csúcsokat érintve ~ c(T_2*) <= c(T_1*)
   (Meg tudjuk tenni, hisz teljes a gráf, de persze itt T_1* kívűli G-beli élt is használunk, de ettől a súly csak csökkeni fog mivel metrikus a gráf!)
5. T_3* := T_2*-ból elhagyva egy élet ~ SOL <= c(T_3*) <= c(T_2*)
   (Így csökken a súlya és T-beli feszítőfát kapunk, amitől a minimális feszítő fa nyilván kisebb egyenlő.)

Tehát: SOL <= c(T_3*) <= c(T_2*) <= c(T_1*) = 2*c(T_0*) = 2*OPT

Éles példa 2-approximációra metrikus gráfon

Teljes, metrikus gráf, egyetlen Steiner ponttal.
T-n belül 2, T és T között 1 az élek költsége.
OPT = n-1 (Csak az S és T közötti éleket vesszük be, |T|=n-1, élek súlya 1).
SOL = 2(n-2) (Csak T közötti éleket vesszük be, ebből n-2 hosszú út elég pl, de súlyaik 2).
$\lim_{T\to\infty} \frac{2(n-2)}{n-1} = 2$ ~ Azaz tudunk találni minden x < 2-re olyan |T|=c számot, amire a fenti példa elrontaná a x-approximációt.

Általános Steiner-fa visszavezetése metrikus Steiner-fára

(G továbbra is egyszerű, összefüggő.)
1. Metrizál G->G': minden él az összekötött két csúcs távolsága lesz az eredeti gráfban ~ OPT'<=OPT
   Pl. Floyd algoritmusa $O(n^3)$ elvégzi a metrizálást.
   (Mivel összefüggő, ezért teljes gráfot kaptunk, a háromszög egyenlőtlenség azért igaz, hisz egy-egy él egy-egy útnak felelt meg az eredeti gráfban, ha létezne a metrizált olyan út, ami egy él helyett gyorsabb lenne, akkor az eredetiben is létezne a kiválaszott legrövidebb út helyett egy rövidebb út, ami ellentmondás.)
   (G'-ben minden él szerepel, ami G-ben, illetve az élek súlya csak csökkenhettek, hisz maga az él is út a két csúcs között, tehát G-ben a minimális Steiner-fa, továbbra is Steiner-fa G'-ben, de előfordulhat, hogy már nem minimális, ezért OPT'<=OPT.)
2. Keres 2-approx F_1' Steiner-fa-t G'-ben. ~ c(F_1')<=2*OPT'
3. F_1 legyen F_1' miután lecseréltünk minden élet a metrizálásnál neki értéket adó legrövidebb út éleire ~ c(F_1) = c(F_1')
4. F_2 legyen F_1-en keresett minimális feszítő fa c(F_2) <= c(F_1)
   (A demetrizálás során egy él többször is bekerülhet vagy feleslegessé válhat.)

Tehát:

5. SOL = $c(F_2)$ <= $c(F_1)$ <= $c(F_1')$ <= 2*OPT' <= 2*OPT

11. A Hamilton-kör probléma visszavezetése az általános utazóügynök probléma k-approximációs megoldására. Közelítő algoritmusok a metrikus utazóügynök problémára, Christofides algoritmusa.

Általános utazó ügynök feladat

Feladat

Adott egy *G* teljes gráf, nemnegatív élsúly függvény.
Keressük a minimális összsúlyú Hamilton-kört. A probléma NP-nehéz.

Utazó ügynök approximáxiója

Nem ismert semmilyen k-appoximációja és nem is létezhet, hacsak nem P=NP.

Hamilton-kör visszavezetése utazó ügynök k-approximációjára

G helyett G' teljes gráf (|G|=n)

$$c(e) = \begin{cases} 1, ha\ e \in G \\ k * n, ha\ e \in \overline{G} \end{cases}$$

G'-ben a lehetséges Hamilton-körök súlyai minimálistól kezdve:
n*1 (n db eredeti élből ⇔ létezik H-kör G-ben)
kn + (n-1)*1 (n-1 db eredeti él, és egy darab kn)
Ha létezik H-kör G-ben, akkor létezik n*1 súlyú H-kör G'-ben, tehát egy k-approximáció algoritmus egy maximum k*n*1 méretű H-kört kell megtalálnia, de k*n*1 <= kn + (n-1)*1, tehát csak a n*1 súlyú H-körök tesznek ennek eleget, ezzel viszont egy H-kört is megtalálnánk.
(Ez igazából k helyett tetszőleges f(n)-re igaz, tehát semmilyen approximációja nem létezhet ennek a problémának.)

Metrikus utazóügynök probléma

Feladat

Adott G teljes gráf, és egy c:E→$\mathbf{R}^+$ élsúlyozás, amire teljesül a háromszög-egyenlőtlenség.
Keressük a minimális összsúlyú Hamilton-kört.
(Ez NP-nehéz, létező él -> 1, nem létező él -> 2 súlyozással visszavezethető Hamilton-körre.)

2-approximációs algoritmus

Algoritmus:

1. F := G minimális feszítő fája
2. E := F éleit duplázva
3. H := E-n keres Euler sétát és a mentén bejár összes csúcsot egyszer (amiben már voltunk azt átugorjuk, mivel teljes a gráf ezért mindig lesz él a következő nem bejárt csúcsba) így kapva egy H-kört

2-approximáció bizonyítása:
Minimális Hamilton-körből elhagyva egy élet egy olcsóbb Hamilton-utat kapunk, viszont a H-út egyben feszítőfa, tehát ennél olcsóbb F, ami egy minimális feszítőfa.
Az él duplázás kétszerezi a költséget, de a levágások már csak csökkentik, tehát:
c(talált H-kör) ≤ c(dupla él m feszítőfa) = 2*c(min feszítőfa) ≤ 2*c(H-út) < 2*c(min H-kör) = 2*OPT

Christofides algoritmus

Algoritmus:

1. F := G minimális feszítő fája
2. minTP : F páratlan fokú csúcsain egy minimális teljes párosítás (G-beli éleket használva)
   (Ez azért jó, mert így a F u TP minden csúcsa páros fokú lesz és így garantáltan lesz Euler-séta benne a drága élkétszerezés nélkül is.)
3. H := F u minTP-n keres Euler sétát és a mentén bejár összes csúcsot egyszer (amiben már voltunk azt átugorjuk, mivel teljes a gráf ezért mindig lesz él a következő nem bejárt csúcsba) így kapva egy H-kört

<u>3/2-approximáció:</u>

A 2-approximációnál ismertett indoklás alapján analóg módon felírható:

c(talált H-kör) ≤ c(F u minTP) <= 3/2*c(min feszítőfa) ≤ 3/2*c(H-út) < 3/2*c(min H-kör) = 3/2*OPT

Csak ezt kéne belátni, hogy c(F u minTP) <= 3/2*c(min feszítőfa), amihez elég, hogy

c(TP) <= ½*c(min feszítőfa):

Egyrészt tudjuk: c(min feszítőfa) <= c(H-út) <= c(min H-kör)

De egy H-kört fel tudunk osztani két diszjunk TP-re úgy, hogy c(TP1) + c(TP2) <= c(min H-kör), azaz

c(TP1) <= ½*c(min H-kör) (feltéve, hogy TP1 a kisebb költségű)

Ezekből már összerakható, hogy

c(minTP) <= c(TP1) <= ½*c(min-Hkör)

(Azért vegyük észre, hogy itt nem a teljes gráf H-köréről van szó, hanem csak a F-ben páratlan fokú csúcsok alkotta részgráf H-köréről, ám ez a metrikusság miatt csak tovább csökkentheti a költséget, növelni nem tudja. Hiszen a minimális költségű H-körben be tudjuk járni csak a kívánt csúcsokat úgy, hogy a metrikusság miatt a költség csökken.)

12. Teljesen polinomiális approximációs séma fogalma. A részösszeg probléma, bonyolultsága. Teljesen polinomiális approximációs séma a részösszeg problémára.

Polinomiális approximációs séma

Definició

∀ ε > 0 : ∃ (1+ ε)-approxmáció egy feladatra
Miben legyen polinomiális?
Bemenet mérete, de akkor lehet pl. $O(2^{1/\varepsilon}*n)$, ami ε csökkenésére (hiba csökkentésére) exponenciálisan nő, ezért bevezetendő:

Teljesen polinomiális approximációs

Ha 1/ε-ben is polinomiáls.
n^{ε} viszont teljesen megfelel, hiszen ε, a hiba mértéke, ha a hiba csökkentésével tudjuk exponenciális mérétkben csökkenti a lépésszámot is, akkor az csak jó.
(Pl. Eukilideszi utazó ügynökre létezik ilyen.)

Részösszeg probléma

Feladat

Adott $A = \{a_1, a_2, ..., a_n\}$ és t.
$\exists\ B \subseteq A : \Sigma b_i = t$?
Speciális eset: partíció probléma, amikor $t = \Sigma a_i/2$, még ez is NP-teljes.

Optimalizálási feladat

Adott $A = \{a_1, a_2, ..., a_n\}$ és t.
Find $B \subseteq A$: max { Σb_i } && $\Sigma b_i \leq t$
A feladat NP-nehéz, mert a részösszeg probléma visszavezethető rá.
(Ha találunk olyan B halmazt, amire $\Sigma b_i = t$, akkor igen a válasz a részösszeg problémára, különben nem.)
A feladat nem NP-beli, mert nem eldöntési probléma, tehát nem is NP-teljes.

Megoldása

Brute force: megviszgálunk minden lehetséges részösszeget
$L_0 = \{0\}$
$L_i' = \{l+a_{i+1} \mid l \in L_i\}$
$L_{i+1} = L_i \cup L_i'$
Kezdjük az első 0 elemből álló részösszegekből.
Ha már ismerjük a első i elemből álló résszösszegeket, akkor ezekhez hozzáadhatjuk az i-dik elemet.
Ekkor az i elemű részösszegek és azokhoz i-dik elemet hozzávett részösszegek kiadják az első i+1 elemből képezhető részösszegeket.
(Nyesési lehetőség: ami túllépte a t-t vagy ismétlödő, azt eldobjuk.)
L_n az összes réssz összeg elő fog állni, a legnagyobb <t-t válasszuk.
(Ha az elemeket növő sorrendben rendezzük és a listákat összefésülésses rendezéssel uniozzuk, akkor L_i rendezett marad így elég a végéről az elsőt kiválasztani, ami már t-től kisebb.)
A listák mérete $2^{|L|}$-re duzzadhat, ezért exponenciális lesz a lépésszám.

Közelítő algoritmusa

A pontos megoldásban lineáris számú iteráció volt, ahol az egyes iterációk lineárisak a lista méretével és elemszámmal is.

A problémát az okozta, hogy a listák hossza exponenciális lehetett, ezért az approximációnk alapötlete, hogy ritkítsuk a lista elemeit minden iteráció után, hogy ne tudjanak nagyon elszaporodni a lehetséges részösszegek, de mindig maradjanak olyan elemek benn, akik jó közelítéssel „képviselni" tudják, ha esetleg épp az optimumot dobtuk el.

Ehhez fontos lesz, hogy a lista végig növekvő sorrendben maradjon, egy δ érték szerint ritkítunk.

Adott l elem, elhagyható minden olyan elem utána, ami kisebb egyenlő, mint (1+δ)l.

(Balról kezdünk el elhagyni elemeket, hogy ha elhagyunk valamit, akkor az már nem tud egy másik elemet képviseiplni.)

(Természetesen az L_n listát nincs értelme ritkítani.)

Az algoritmus akkor adhat nem pontos eredményt, ha azt valamikor kitöröltük a listából ám ekkor léteznie kellett annak egy képviselőjének, ami max (1+δ) hibát jelent, ám a képviselőt képviseltethettük egy másik elemmel, ez maximum n-szer történhetett meg, azaz a hiba maximum $(1+\delta)^n$ lehet.

Teljesen polinomiális approximációs sémája

Azt fogjuk belátni, hogy δ=ε/2n választással a hiba faktor (1+ ε) a lépésszám pedig polinomiális n, logt (bemenet) és 1/ε-ben is (teljesen polinomiális).

<u>(1+ ε) multiplikatív hiba:</u>

Láttuk, hogy a hiba max $(1+\delta)^n = (1+\varepsilon/2n)^n$, lássuk be, hogy ez <= 1+ ε.

$$\left(1+\frac{\varepsilon}{2n}\right)^n = \sum_{i=0}^{n}\left(\left(\frac{\varepsilon}{2n}\right)^i * 1^{n-i} * \binom{n}{i}\right) \le \sum_{i=0}^{n}\left(\left(\frac{\varepsilon}{2n}\right)^i * n^i\right) \le \sum_{i=0}^{n}\left(\frac{\varepsilon}{2}\right)^i = 1+\sum_{i=1}^{n}\left(\frac{\varepsilon}{2}\right)^i \le 1+\varepsilon * \sum_{i=1}^{n}\frac{1}{2^i} \le 1+\varepsilon$$

Binomiális tétel

(n alatt i) felülről becsülhető n^i-vel

Egyszerűsítünk n^i-vel

Első tagot kifejtjük szumma elé

ε<=1, tehát ε>=ε^i, ε szorzót ekkor ki tudjuk hozni szumma elé

<u>Polinomiális lista méret:</u>

A lista első eleme 0, a következő legalább 1, ez után viszont két elem közti arány min (1+ ε/2n) (következő/előző), hisz különben ritkítottuk volna az adott elemet., így az i-dik elem legalább

$(1+\varepsilon/2n)^{i-2}$ értékű (-2: első és második elem miatt).

A t-nél nagyobbakat viszont eldobjuk, tehát ennél kisebb minden elem értéke:

$$(1+\frac{\varepsilon}{2n})^{i-2} \le t \sim vegy\text{ünk } 1+\frac{\varepsilon}{2n}\ akapú\ logaritmust$$

$$i-2 \le log_{\left(1+\frac{\varepsilon}{2n}\right)}t = \frac{\ln t}{ln\left(1+\frac{\varepsilon}{2n}\right)} \sim lemmát\ alkalmazzuk\ jobb\ oldalt$$

$$i-2 \le \frac{\ln t}{\frac{\frac{\varepsilon}{2n}}{1+\frac{\varepsilon}{2n}}} = \frac{2n\left(1+\frac{\varepsilon}{2n}\right)lnt}{\varepsilon}$$

Tehát ezek szerint a listában szereplő i-dik elemnek a sorszáma maximum az adott kifejezés lehet, ami viszont n-ben, lnt-ben (bemenet) és 1/ε polinomiális.

<u>Lemma:</u>

$$ln(1+X) \ge \frac{X}{1+X} \quad X \ge 0$$

$$\left(ln(1+X)-\frac{X}{1+X}\right)' = \frac{1}{1+X}-\frac{(1+X)-X}{(1+X)^2} = \frac{1}{1+X}-\frac{1}{(1+X)^2} > 0$$

Tehát a kettő különbségének deriváltja pozitív, azaz a különbségük monoton növő, ami már bizonyítja az egyenlőtlenséget.

13. Ütemezési feladatok típusai. Az $1|prec|C_{max}$ és az $1||\Sigma C_j$ feladat. Approximációs algoritmusok a $P||C_{max}$ feladatra: listás ütemezés tetszőleges sorrendben, éles példa tetszőleges számú gép esetére; listás ütemezés LPT sorrendben (biz. nélkül), éles példa tetszőleges számú gép esetére. Approximációs algoritmus a $P|prec|C_{max}$ feladatra (biz. nélkül), példák: az LPT sorrend, illetve a leghosszabb út szerinti ütemezés sem jobb, mint (2−1/m)-approximáció. A $P|prec,p_i = 1|C_{max}$ feladat, Hu algoritmusa (biz. nélkül).

Ütemezési feladatok alapjai

- Definíciók:
  - J_i munkák (job)
  - p_i megmunkálási idő (processing time)
  - ω_i súly (weight)
  - d_i határidő (due date)
  - r_i rendelkezésre állási idő (release time) ~ mikortól kezdve lehet
  - C_i befejezési idő (completion time)
- egy adott időben egy gépen egy munka folyik
- célfüggvényre optimalizálunk (pl.: max(Ci): teljes átfutási idő)

Ütemezési feladatok osztályozása

Az ütemezési feladatok egy $\alpha|\beta|\gamma$ hármassal írhatók le, ahol

- α a gépek száma
  - 1: 1 gép
  - p_m: m párhuzamosan futó gép
  - P: nem rögzített számú párhuzamosan futó gép.
    $P|\beta|\gamma$ az $\{1|\beta|\gamma, P_2|\beta|\gamma, P_3|\beta|\gamma, ...\}$ feladatokat tartalmazó osztály neve.
- β az infók halmaza az ütemezésről
  - precedencia: adott egy irányított gráf (DAG), ami megkötéseket tartalmaz arra nézve, hogy egy adott munka elkezdéséhez mely munkákat kell előbb befejezni
  - r_j: adottak a release time-ok, azaz hogy melyik job mikortól áll rendelkezésre
  - p_j: adottak a megmunkálási idők, mennyit vesz igénybe az adott munka elvégzése
- γ a függvény, ami szerint optimalizálunk
  - $\min C_{max} = \max(C_j)$: az utolsó munka befejezési ideje
  - $\min \sum C_j \sim \sum C_j/n$: a munkák átlagos befejezési ideje
  - $\min \sum C_j - d_j$: a késések összege

$1||C_{max}$

Optimális megoldást ad minden olyan megoldás, melynél a gép folyamatosan dolgozik.
Ennek értéke a munka elvégzési idők összege.

$1|prec|C_{max}$

A feladatokat topologikus sorrendben adogatjuk a gépnek.
Ez optimális, mert folyamatosan foglalt a gép és értéke továbbra is a munka elvégzési idők összege.

$1||\sum C_j$

SPT (Shortest Processing Time)

Munkaigény szerint növekvő sorrendben adogatjuk a feladatokat a gépnek. $p_1 \leq p_2 \leq ... \leq p_n$.
Az SPT ütemezés optimális, mert véges sok sorrend létezik és egy nem SPT ütemezés javítható. Ha az ütemezés J_1, ..., J_n sorrendben veszi a feladatokat, ahol valamelyik i-re $p_i > p_{i+1}$, akkor a két feladatot megcserélve $\sum C_j$ nő.
(p'_{i+1} befejezési ideje ugyanaz, mint p_{i+1}, de p'_i-jé csökken $p_i - p_{i+1}$ értékkel.)

$P||C_{max}$

Bonyolultság

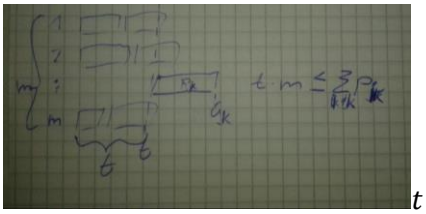
A $2||C_{max}$ feladat NP nehéz, mert visszavezethető rá a partíciós probléma. $P||C_{max}$ is NP nehéz, mert speciális esetként tartalmazza $2||C_{max}$-ot.
(Legyenenek a munkaidők a particiós probléma bemenetei, ha ezek optimális megoldása
$C_{max} <= \Sigma p_i/2$, akkor particionálható ~ lényegében itt csak egyenlőség lehet és úgy, hogy mindkét gép folyamatosan dolgozik és egyszerre végez.)

LS listás ütemezés (2-1/m)

Algoritmus:
A munkáknak adott egy sorrendje, ha felszabadul egy gép akkor azonnal kezdi a listában a legelső még nem elkezdett munkát.
(2-1/m) approximáció:
Könnyen látható, hogy a listás ütemezés polinomidőben elkészíthető.
Az approximációs faktor igazolásához legyen:

- J_k utoljára végződő munka (nem ugyanaz, mint az utolsóként elkezdett!)
- t J_k megkezdésének ideje

$C_{max} = t + p_k$
Mivel minden gép folyamatosan foglalt legalább a t időpillanatig (különben az LS algoritmus szabálya szerint a J_k munkához valamely gép már előbb hozzákezdett volna), így:

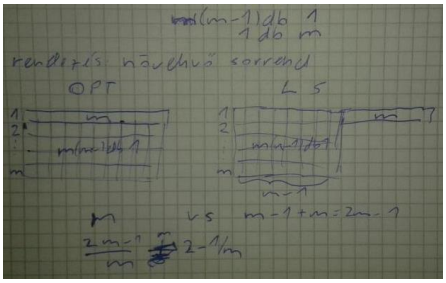


$t \leq \frac{1}{m}\left(\sum_{j \neq k} p_j\right)$

Jelölje C^*_{max} az optimális teljes átfutási időt. Ekkor:

$$C^*_{max} \geq \frac{1}{m}\left(\sum_{1}^{n} p_j\right) \qquad C^*_{max} \geq max_j\{p_j\}$$

Ezek alapján levezethető, hogy:
$C_{max} = t + p_k \leq \frac{1}{m}\left(\sum_{j \neq k} p_j\right) + p_k = \frac{1}{m}\left(\sum_j p_j\right) + \left(1 - \frac{1}{m}\right) p_k \leq C^*_{max} + \left(1 - \frac{1}{m}\right) C^*_{max} = \left(2 - \frac{1}{m}\right) C^*_{max}$
Tehát sikerült igazolni az approximációs faktor helyességét.
Éles példa:



LS-LPT (Longest Processing Time)

Algoritmus:

LS ütemezés megmunkálási idő szerint csökkenő sorrendbe rendezve.

(Ezzel a fenti éles példa kiküszöbölése a cél: ne a végére maradjanak a nagy munkák, hisz őket nehezebb egyenletesen beosztani, előbb kezdjük el őket és a szabad helyerek „betuskoljuk” a végén a kicsi munkákat.)

Éles példa ($(\frac{4}{3} - \frac{1}{3m})$):

2db:

2m-1, 2m-2, ... m+1-ig

3db:

m

OPT = 3m:

A 2db-sokból (m-1) db 3m párt csinál, a 3db m-t egy 3m-mé teszi össze

Így végül m db 3m-t kap, amit egy-egy gépre téve 3m alatt végez

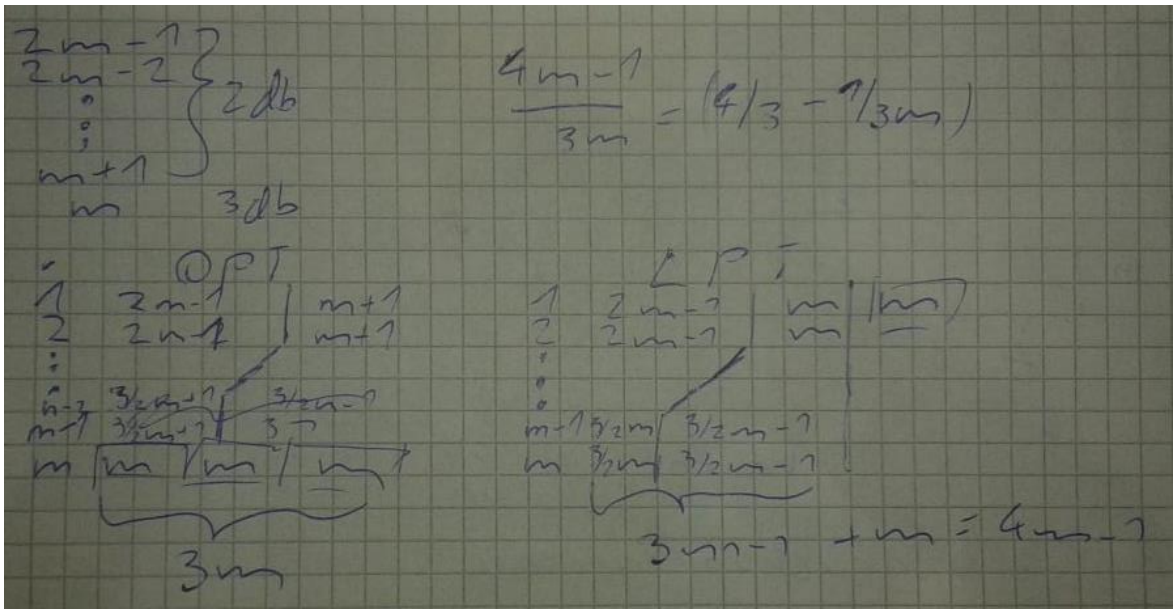
LPT = 4m-1:

Először elkezdi feltölteni a gépeket fenntről lefelé a leghosszabbakkal: 2m-1,...,3/2m

(Hisz m gép van, de mindegyikből 2db van, ezért csak „-1/2m”-vel csökken a számaik)

Majd a lenntről felfelé érnek véget és tölti fel a gépeket újra a 3/2m-1,...,m-ig, de egy db m még hátra marad

Egy-egy gépen (2m-1)+(m), (2m-2)+(m+1),...,(3/2m)+(3/2m-1) feladatpárok futnak, ezek elvégézse egységesen 3m-1, de még hátra van egy m hosszú így összeségében 4m-1 lesz.



Probléma a listás ütemezéssel

2 munka, 1-1 elvégzéshez szükséges időegység (1-es sebesség esetén), 2 gép (1db 1-es, 1db 10-es sebességű)

Listás ütemezéssel eredmény 1.1, viszont optimális 2/10.

(mindkét munkát 10-es sebességű gép végzi el.)

P|prec|C_{max}

2-1/m approximáció

LS ütemezést kell alkalmazni.

LPT éles példa

m(m-1) db 1

1 db ½

1 db m-½

Precedencia: ½ -> m-½

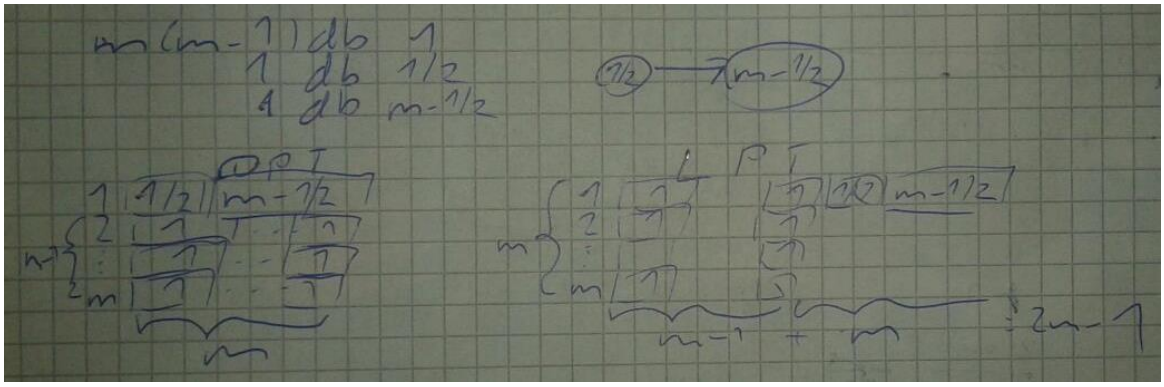
Lényegében azzal romlik el az LPT itt, hogy egy hosszabb munkát beelőzhet egy rövidebb, ha egy még rövidebb munka befejezésétől függ.

LPT 2m-1:

m(m-1)db 1 le fog futni párhuzamosan m-1 idő alatt m gépen, majd lefut a ½ -> m-½ m idő alatt.

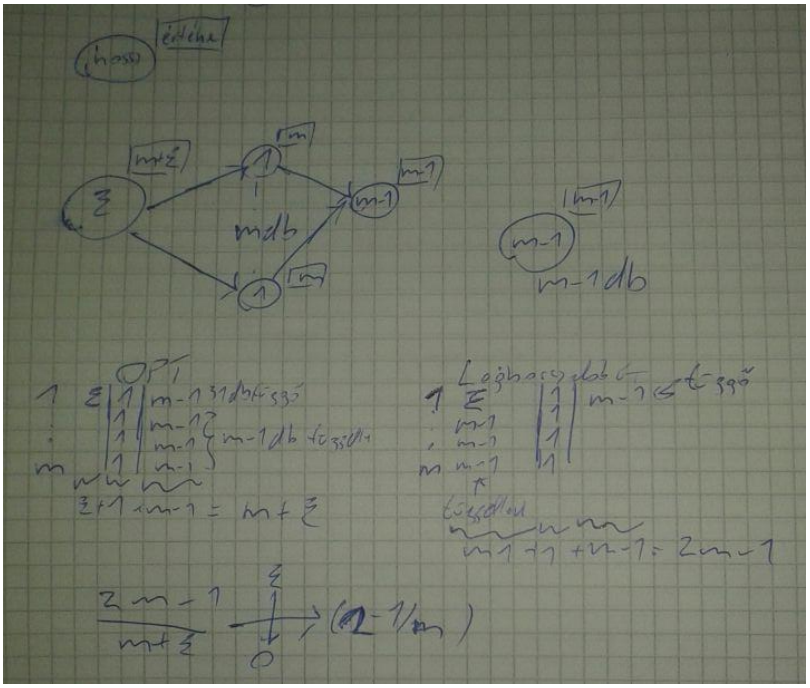
OPT m:

Egyik gép futtatja a ½ -> m-½, a másik m-1 pedig m db 1-est, így egész lefut m alatt.



Leghosszabb út szerinti sorrend éles példa

Az LPT hibájából tanulva gondolhatnánk arra, hogy akkor a leghosszab helyett azt válasszuk, amiből a leghosszabb munkaút indul (precedencia gráfon, ami DAG, ezért pol időben lehet ezt megtalálni).



A job oldali ábra csalóka, vagy akár rossz. Az epszillon végeztével több egyest is elkezdhet elvégezni, de az m -1 időpontig max m-2-t tud elvégezni, ezért lesz olyan 1-es, amit akkor kezd el és így a függő m-1-t csak az m-dik pillanatban tudja elkezdeni, de nem egyszerre fognak menni az 1-esek, hisz listás ütemezésnél nem lehet üresen egy gép, ha van végezhető feladat.

Tehát itt abba a csapdába esik az algoritmus hogy elkezdi futtatni a hossú m-1-ket ahelyett hogy üresen hagyná az automatákat és megvárná míg lefutna az E. Így végül az 1->m-1 hosszabb feladat száll egy rövidebb m-1 feladat szállra vár, hisz a listás ütemezés nem engedi meg azt, hogy az egyik automata üresen álljon.

$P|prec, p_j=1|C_{max}$

Bonyolultság

A feladatosztály bonyolultsága

- $P|prec, p_j=1|C_{max}$ NP-nehéz
- $P2|prec, p_j=1|C_{max}$ Coffman és Graham algoritmus ad optimális megoldást
- $P_m|prec, p_j=1|C_{max}$ bonyolultsága ismeretlen
- $P|prec, p_j=1$, in-tree$|C_{max}$-re a Hu-algoritmus polinom időben optimális megoldást ad.

Hu algoritmusa

A $P|p_i=1, prec|C_{max}$ feladatra ad megoldást, ha a precendenciagráf *in-tree (be-fenyő)* tulajdonságú.

1. A precedencia gráf minden csúcsához határozzuk meg a gyökérbe vezető út hosszát.
2. Alkalmazzuk a listás ütemezést úthossz szerint csökkenő sorrendben.

(Az *in-tree (be-fenyő)* egy olyan irányított gyökeres fa, aminek az élei a gyökér felé vannak irányítva.)

14. Globális és lokális élösszefüggőség és élösszefüggőségi szám fogalma, Menger irányítatlan gráfokra és élösszefüggőségre vonatkozó két tétele (biz. nélkül). λ(G) meghatározása folyamok segítségével négyzetes és lineáris számú folyamkereséssel. λ(G) meghatározása összehúzások segítségével, Mader tétele, Nagamochi és Ibaraki algoritmusa (biz. nélkül). Minimális méretű 2-élösszefüggő részgráfok keresése. A probléma NP-nehézsége, Khuller-Vishkin algoritmus (biz. nélkül).

Lokális élösszefüggőség

Egy $G = (V, E)$ gráfban $u, v \in V$ pontok közötti lokális élösszefüggőség ($\lambda(u, v)$) az *u* és *v* pontok közötti éldiszjunkt utak száma. $\lambda(u, v) = \min\{d(X): X \subset V, u \in X, v \notin X\}$, vagyis az *u*, *v* pontokat elválasztó ponthalmaz minimális fokszáma.

Gráf élösszefüggőségi száma

A lokális élösszefüggőségi számok közül a legkisebb, $\lambda(G) = \min\{\lambda(u, v): u, v \in V, u \neq v\}$.

Összefüggőség kiszámítása

G-ből készítünk egy hálózatot egységnyi élsúlyozással, és keressük a maximális folyamot, vagyis a minimális vágást. Két pont összehúzásával a vágás nem csökkenhet, és csak akkor nőhet, ha az összes minimális vágás a két pont között halad át.

Vegyük $\lambda(G) = \min\{\lambda(u, v), \lambda(G_{u,v})\}$, ahol $G_{u,v}$ a *G*-ből az *u* és *v* pontok összehúzásával kapott gráf. Ezt n – 1-szer kiszámolva megkapjuk *λ(G)*-t

A gráf pontjainak egy $(v_1, v_2, \dots, v_n)$ sorrendje max-vissza sorrend, ha $i < j$ esetén $d(v_i, V_{i-1}) \geq d(v_j, V_{i-1})$, ahol $d(v_i, V_j)$ a v_i és $\{v_1, \dots, v_j\}$ pontok közötti fokszám.

Ha *u* és *v* a max-vissza sorrend szerinti utolsó két pont, akkor $\lambda(u, v) = d(v)$. Ha tehát minden iterációban az összehúzandó párt a max-vissza sorrend utolsó két pontjaként választjuk, akkor a $\lambda(u, v)$ egyszerűen *v* fokszáma lesz. (Ekkor eltekinthetünk a folyam számolásától)

Összefüggőségek

Def.: Egy $G = (V, E)$ gráfban $u, v \in V$ pontok közötti lokális élösszefüggőség ($\lambda(u, v)$) az *u* és *v* pontok közötti éldiszjunkt utak száma.

Def.: Egy $G = (V, E)$ gráfban $u, v \in V$ pontok közötti lokális összefüggőség ($\kappa(u, v)$) az *u* és *v* pontok közötti belsőleg pontdiszjunkt utak száma.

Def.: Egy G gráf k-összefüggő (pontösszefüggő), ha létezik k+1 csúcsa és bármely maximum k-1 darab csúcsát elhagyva összefüggő marad.

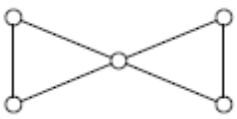
Def.: Egy G gráf k-élösszefüggő, ha bármely maximum k-1 darab élét elhagyva összefüggő marad. A k-összefüggőségből következik a k-élösszefüggőség.

Def.: A $\lambda(G)$ élösszefüggőségi szám az a maximális k érték, hogy a G gráf még k-élösszefüggő.

Def.: A $\kappa(G)$ összefüggőségi szám az a maximális k érték, hogy a G gráf még k-összefüggő.

Általánosan igaz, hogy: $\lambda(G) \geq \kappa(G)$

Példa arra, hogy a gráf 2-élösszefüggő, de nem 2-összefüggő:



Menger-tétel

Def.: Egy F élhalmaz lefogó élhalmaz, ha a G (irányított vagy irányítatlan) gráf F élhalmaza lefog minden u-v utat, ha a G-F gráfban nem létezik u-ból v-be (irányított) út.

1. Ha G egy irányított gráf, $s, t \in V(G)$, akkor az s-ből t-be vezető éldiszjunkt irányított utak maximális száma megegyezik az összes irányított s – t utat lefogó élek minimális számával.
2. Ha G egy irányított gráf, $s, t \in V(G)$ két nem szomszédos pont, akkor az s-ből t-be vezető, végpontoktól eltekintve pontidegen irányított utak maximális száma megegyezik az összes irányított s – t utat s és t felhasználása nélkül lefogó pontok minimális számával.
3. Ha G egy irányítatlan gráf, $s, t \in V(G)$, akkor az s-ből t-be vezető élidegen irányítatlan utak maximális száma megegyezik az összes irányítatlan s – t utat lefogó élek minimális számával.
4. Ha G egy irányítatlan gráf, $s, t \in V(G)$ két nem szomszédos pont, akkor s-ből t-be vezető pontidegen irányítatlan utak maximális száma megegyezik az összes irányítatlan s – t utat s és t felhasználása nélkül lefogó pontok minimális számával.

A téma szempontjából lényeges állítás: Az u és v közötti éldiszjunkt utak maximális száma megegyezik az u és v közötti utakat lefogó élek minimális számával.

λ(G) meghatározása folyamok segítségével

Ha a gráfot egy hálózati folyamként értelmezzük, ahol minden él kapacitása egy, akkor *λ(u,v)* épp a maxfolyam probléma.
Ezt a problémát minden pontpárra megoldhatjuk, azaz összesen O(n^2) db folyamot kell keresnünk. Megfigyelés, hogy általában *n*-1 folyamprobléma megoldása is elég.
Maximális folyam keresésére használhatjuk az Edmonds-Karp algoritmust. Ez az eredeti Ford-Fulkerson algoritmus továbbfejlesztése. A Ford-Fulkerson algoritmus irracionális kapacitásértékek előfordulása esetén nem állt le garantáltan, míg az Edmonds-Karp igen.

Működése:

1. Inicializálás: kezdeti folyam létrehozása (pl. ekvivalens 0 érték minden élre)
2. Javító folyam keresése. Ford-Fulkerson esetén ennek módszere mindegy, Edmonds-Karp esetén ennek a legrövidebb lehetséges javító folyamnak kell lennie, ami pl. szélességi kereséssel megtalálható.
3. Ha találtunk javító folyamot, akkor annak értékével megnöveljük az oda-élekre a folyamértéket, és lecsökkentjük a vissza-élekre a folyamértéket (ha van ilyen). Majd folytatjuk a 2. lépéstől.
   Ha nincs megfelelő javító folyam, leállunk: találtunk egy maximális folyamot.

Lépésszáma O(|*V*|*|*E*|^2).

A segédgráf létrehozása:

Az irányított G gráfon adott f folyamhoz definiálunk egy G_f irányított gráfot:
Legyen V(G_f)=V(G) és G_f-ben fusson egy irányított él x-ből y-ba, ha vagy $(x, y) \in E(G)$ és f(x,y)<c(x,y) (f: folyam értéke, c: élek kapacitása) vagy $(y, x) \in E(G)$ és $f(x, y) > 0$. Könnyen látható, hogy ha
G_f-ben van egy irányított út s-ből t-be, akkor az ennek az útnak megfelelő élek G-ben épp egy javító utat adnak az f folyamra nézve. Ha pedig van javító út G-ben, akkor lesz irányított út s-ből t-be
G_f-ben.
Így tehát elegendő megnézni, hogy az így kapott G_f gráfban van-e s-ből t-be út. Ezt viszont BFS algoritmussal könnyen eldönthetjük.

Lineáris számú folyamkeresés:

$\lambda(G) = \min\{\lambda(a, v): v \in V - a\}, ahol\ a \in V\ (kötött)$
Ekkor n-1 folyamproblémát kell megoldani. Lépésszáma O(|*V*|*|*V*|^3)= O(|*V*|^4).

Biz.:

A négyzetes számú folyamkereséshez képest szűkebb halmazon vizsgálódunk, ezért azt kell garantálnunk, hogy bármely a csúcsból kiindulva a $\lambda(a,v)$ értékek között lesz $\lambda(G)$ értékű. Legyen $k = \lambda(G)$. Nyílván minden $b \in V(G) - a$ csúcsra igaz, hogy:
$k = min_{u,v \in V(G), u \neq v} \lambda(u,v) \leq min_{v \in V(G)-a} \lambda(a,v) \leq \lambda(a,b)$ azaz $\lambda(a,b) \geq k$.
Hagyjunk el G-ből k élt úgy, hogy szétessen, ekkor a kapott részgráfnak legalább két komponense lesz. Mivel a rögzített, válasszuk ki b-t úgy, hogy a és b külön komponensben legyenek, ekkor $\lambda(a,b) \leq k$. $\lambda(a,b) \geq k$ miatt $\lambda(a,b) = k = \lambda(G)$, azaz lesz olyan b csúcs, ahol a $\lambda(G)$-nek megfelelő minimum felvétetik.

Mader tétel

Minden G gráfban létezik olyan $u, v \in V(G)$ csúcspár, hogy $\lambda(u,v) = d(v)$, ahol d(v) a v pont fokszáma.
Ha *u* és *v* a max-vissza sorrend szerinti utolsó két pont, akkor $\lambda(u,v) = d(v)$. Ha tehát minden iterációban az összehúzandó párt a max-vissza sorrend utolsó két pontjaként választjuk, akkor a $\lambda(u,v)$ egyszerűen d(*v)* fokszáma lesz. Így eltekinthetünk a folyam számolásától.

Nagamochi és Ibaraki tétele

1) $\lambda = \infty$
2) készítsük el a gráf max-vissza sorrendjét. Ha $d(v_n) < \lambda$, akkor legyen $\lambda = d(v_n)$.

Ha még legalább három pontú a gráf, akkor húzzuk össze a max-vissza sorrend szerinti utolsó két pontot, GOTO 2. Egyébként STOP

Minimális méretű 2-élösszefüggő részgráf keresése

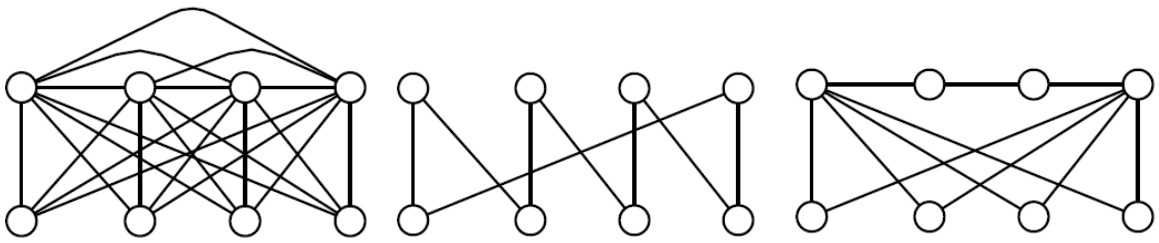
Adott egy *G* 2-élösszefüggő irányítatlan gráf, minden él költsége 1, és $r(u,v) = 2$ (élösszefüggési követelmény; ez a Hamilton-kör általánosítása: „dupla Hamilton-kör"). Feladat egy minimális élszámú 2-összefüggő feszítőgráf keresése *G*-ben. Látható, hogy az optimum értéke $|V|$, ha G-ben van Hamilton-kör. A feladat NP-nehéz, ezért nincs hatékony algoritmus, de közelítés van.

Khuller-Vishkin

Mélységi bejárás (*T*), közben építjük a megoldást: *E'*. *T* éleit bevesszük *E'*-be. Továbbá, ha az algoritmus *v* pontból visszalép egy teljesen bejárt részgráfból, akkor a *T*(*v*)-ből azt a kilépő élet hozzáadjuk *E'*-höz, ami nincs *T*-ben és a *T*(*v*)-n kívüli pontját először érte el a keresés.
$k = \frac{3}{2}$-approximációs.

Példa 1.

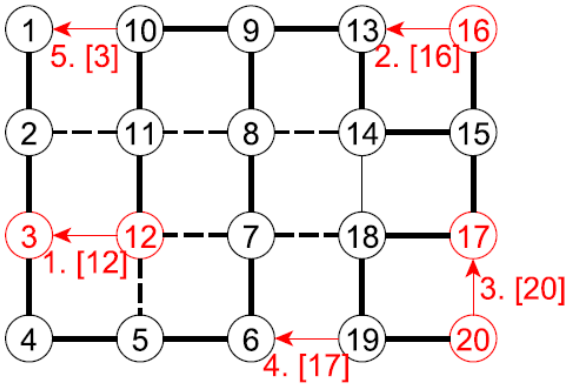
Az éles példa azt jelenti, hogy olyan példa, amikor az approximációs algoritmus a lehető legrosszabb becslést adja (itt pl. az optimum 3/2-szeresét), ami mutatja, hogy a becslés "éles", tehát nem javítható.
Pl. egy olyan gráf, amikor egy teljes páros gráfot pakolunk össze úgy, hogy az egyik komponens *n* darab izolált pont, a másik pedig egy *n* csúcsú teljes gráf. Mondjuk egy $K_{4,4}$ és ennek az egyik osztályába felveszünk még 6 élet.



vizsgálandó gráf optimális megoldás kapott megoldás

Az optimum ilyenkor a Hamilton-kör hossza, tehát 2**n* ($K_{4,4}$-nál 8 él). Az algoritmus pedig úgy lesz "rossz", hogy a mélységi kereséssel először bejárjuk az egyik oldalon lévő teljes részgráfot, ez *n*-1 él, és utána minden, a másik oldalon lévő ponthoz fel kell venni 2 élet, tehát 2*n*-et. Ez összesen 2*n*+*n*-1 él (itt: 11).
Összesítve, az approximációs faktor: $\frac{3n-1}{2n}$, így nem lehet jobb, mint 3/2.

Példa 2.



A csúcsok sorszámai a mélységi bejárás során kapott mélységi számok. A vastag fekete élek jelölik a mélységi bejárás útját, a vékony piros élek a visszalépéskor hozzáadott élek (a feliratuk a behúzás sorszáma, szögletes zárójelben a csúcs, amelyből történő visszalépéskor behúztuk), a szaggatott élek az $E \setminus E'$-beli élek.