Bonyolultságelmélet gyakorlat – 11
Approximálás II.

1. Feladat Algoritmus a MAX2SAT problémára:

MAX2SAT

- **Input**: egy 2CNF, klózonként két különböző változóval.
- **Output**: olyan értékadás, ami a *lehető legtöbb klózt* kielégíti.

Ez is **NP**-teljes eldöntési változattal bíró probléma (annak ellenére, hogy a 2SAT **P**-ben van).

Egy *randomizált* approximáló algoritmus:

- Minden változónak adjunk $\frac{1}{2} - \frac{1}{2}$ eséllyel 0 vagy 1 értéket.

Akkor ez az algoritmus *várható értékben* kielégíti az összes klóz $\frac{3}{4}$-ét. Tehát az optimumnak is kielégíti $\frac{3}{4}$-ét, így (maximalizálási probléma: az optimumot osztjuk az algoritmus eredményével) egy $\frac{4}{3}$-approximáló randomizált algoritmussal van dolgunk.

A várható érték fölé dobni viszont nem is olyan könnyű. Ezért jó, hogy ezt az algoritmust teljesen **derandomizálni** tudjuk:

Egyszerűen sorban adjuk a változóknak az értéket, mindig megyünk a nagyobb várható érték felé, de valójában sosem dobunk randomot. Ez most a következőképp szól, ha az input formulában az $x_1, \ldots, x_n$ változók vannak:

- Ciklusban értékadunk előbb x_1-nek, majd x_2-nek, …, x_n-nek.
- Az x_i értékét a következőképp határozzuk meg: kiszámolunk az $x_i = 0$ és az $x_i = 1$ értékadás mellett is egy-egy értéket, és amelyik esetben nagyobbat kaptunk, azt választjuk (egyenlőségnél pedig mindegy).

  Az értéket úgy kapjuk, hogy minden klózhoz rendelünk egy súlyt, és ezeket a súlyokat összegezzük. Egy klóz súlya

  - 4, ha már van benne 1 értékű literál;
  - 0, ha nincs és már minden változó kapott értéket (tehát csak az $x_1, \ldots, x_i$ változók szerepelnek a klózban);
  - 2, ha nincs és még egy változó van a klózban, aki nem kapott értéket (tehát egy x_j, $j > i$ szerepel még)
  - 3, ha a klózban még egyik változó se kapott értéket.

Ez minden. Nyilván lehet gyorsításokat eszközölni (pl. a súlyösszegek közti viszonyt nem befolyásolják azok a klózok, amikben x_i nem is szerepel, így ezekre nem kell súlyt számolni stb.), és persze illik is, de a lényeg ez marad.

Példa:

$(\mathbf{x_1} \vee \neg \mathbf{x_2}) \wedge (\neg \mathbf{x_1} \vee \mathbf{x_2}) \wedge (\mathbf{x_1} \vee \mathbf{x_3}) \wedge (\mathbf{x_2} \vee \neg \mathbf{x_3}) \wedge (\neg \mathbf{x_2} \vee \neg \mathbf{x_4}) \wedge (\mathbf{x_2} \vee \neg \mathbf{x_5}) \wedge$

$\wedge (\neg \mathbf{x_2} \vee \mathbf{x_5}) \wedge (\mathbf{x_2} \vee \mathbf{x_5}) \wedge (\neg \mathbf{x_2} \vee \neg \mathbf{x_5}) \wedge (\neg \mathbf{x_3} \vee \mathbf{x_4}) \wedge (\mathbf{x_3} \vee \neg \mathbf{x_4}) \wedge (\neg \mathbf{x_3} \vee \neg \mathbf{x_4})$

Megoldás

Első iteráció (x_1 kap értéket):

- $x_1 = 0: \ 2+4+2+3+3+3+3+3+3+3+3+3 = 35$
- $x_1 = 1: \ 4+2+4+3+3+3+3+3+3+3+3+3 = 37$

$\Rightarrow \ x_1 = 1$

Második iteráció (x_2 kap értéket):

- $x_2 = 0: \ 4+0+4+2+4+2+4+2+4+3+3+3 = 35$
- $x_2 = 1: \ 4+4+4+4+2+4+2+4+2+3+3+3 = 39$

$\Rightarrow \ x_2 = 1$

Harmadik iteráció (x_3 kap értéket):

- $x_3 = 0: \ 4+4+4+4+2+4+2+4+2+4+2+4 = 40$
- $x_3 = 1: \ 4+4+4+4+2+4+2+4+2+2+4+2 = 38$

$\Rightarrow \ x_3 = 0$

Negyedik iteráció (x_4 kap értéket):

- $x_4 = 0: \ 4+4+4+4+4+4+2+4+2+4+4+4 = 44$
- $x_4 = 1: \ 4+4+4+4+0+4+2+4+2+4+0+4 = 36$

$\Rightarrow \ x_4 = 0$

Ötödik iteráció (x_5 kap értéket):

- $x_5 = 0: \ 4+4+4+4+4+4+0+4+4+4+4+4 = 44$
- $x_5 = 1: \ 4+4+4+4+4+4+4+4+0+4+4+4 = 44$

$\Rightarrow \ x_5 = 0$

2. Feladat Algoritmus a Metrikus Tsp probléma optimalizálási változatára:

Metrikus Tsp

- **Input**: városok közti távolságok, melyek teljesítik a háromszög-egyenlőtlenséget: $d(a,b) + d(b,c) \geq d(a,c)$ bármelyik a, b, c városokra.

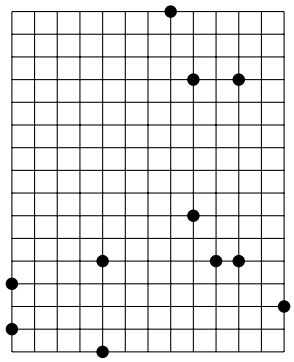
  (Ennek pl egy esete, mikor ténylegesen a síkon vannak leszórva a pontok és távolságuk a szokásos euklideszi távolság.)

- **Output**: minimális költségű (össztávolságú) körút

Algoritmus:

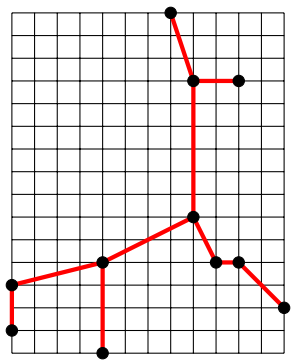
- kiszámítjuk (pl Prim algoritmusával) az input halmaz egy minimális feszítőfáját:
  - kiindulunk egy egyelemű S halmazból ($S \in V$)
  - bekötjük ciklusban a legkisebb költségű $S \to V \backslash S$ közti éllel $V \backslash S$ még egy csúcsát
- ennek valamelyik pontjából, mint gyökérből preorder bejárás szerint kiírjuk a csúcsokat.

Példa: ha a csúcsok koordinátái a $2D$ síkon $(0,1)$, $(0,3)$, $(4,0)$, $(4,4)$, $(7,15)$, $(8,6)$, $(8,12)$, $(9,4)$, $(10,4)$, $(10,12)$, $(12,2)$

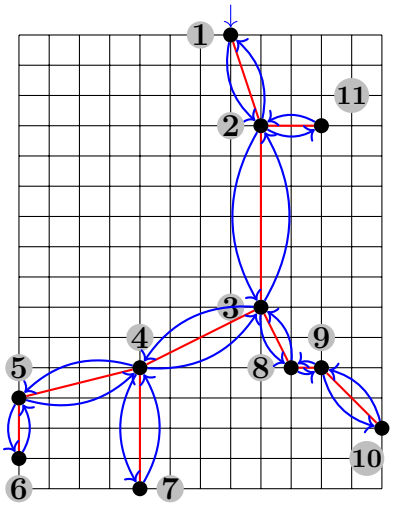


Megoldás

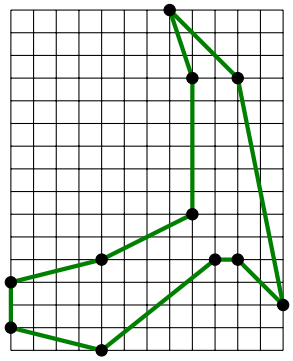
Először is ezen pl a $(7,15)$ pontból indítva a Prim algoritmust (mely mindig a még bekötetlen csúcsok közül a legrövidebb éllel beköthetőt választja és jelöli meg bekötöttként) a következő feszítőfát (piros) kapjuk:



amiből a visszaadott körút, ha megint a $(7, 15)$-ből mint gyökérből indítjuk a preorder bejárást:



amiből az elérési sorrend alapján (azaz a *már meglátogatott* csúcsokat kihagyva) a következő körsétát kapjuk:



Ez a körút tehát legfeljebb kétszer olyan hosszú, mint az optimális.