Tanislav Alexia
grupa 232

11.04.2025

Algoritmi aproximativi

KNAPSACK

Avem $S = \{s_1, s_2, \dots s_n\}$ și K = nr. natural, $K \geq s_i$, $i = \overline{1,n}$

Trebuie să alegem un subset din S a.î. suma elementelor să fie maximă și $\leq K$. Fiecare element poate fi luat o singură dată!

a) Seamănă cu knapsack 0/1 (cap weight = val)

K = capacitatea rucsacului

$s_i = \begin{cases} i = \text{obiectul}, \forall i = \overline{1,n} \\ s_i = \text{greutatea lui, val. lui.} \end{cases}$

Ne folosim de metoda programării dinamice:

Avem un vector $v = \underbrace{\{True, False, False \dots False\}}_{K+1}$, unde fiecare

$v[j]$ reprezintă dacă acea sumă se poate face sau nu din elementele din S. ($v[0]$ = True pt. că ai o elem. care pot face 0)

pentru fiecare elem. din S parcurgem v de la dreapta/coadă la stg/cap și verificăm dacă $v[j - s_i]$ == True => $v[j]$ = True

exemplu: $v[3]$ = True pt că avem un $s_i = 3$

$v[10] = \begin{cases} \text{True, dacă avem și un } s_i = 7 \\ \text{False, rămâne așa dacă nu avem } v[7] = \text{True} \end{cases}$

Acesta e parcurs până la $i = n$, iar răsp. final este ultimul elem. din v cu val True.

Complexitate $\begin{cases} \text{timp}: O(n \cdot K) \\ \text{spațiu}: O(n \cdot K) \end{cases}$

Este optimă soluția pt. că explorează toate sumele și garantează suma maximă.

b) timp $O(n)$ și spațiu $O(1)$

! ne trebuie o sumă $\frac{k}{2}$ max cel puțin

Găsim cel mai mare elem. $\le k$ sau facem o sumă temporară cât timp nu depășim k.

Când suma temp $> k \Rightarrow$ mai mare el decât $\frac{k}{2}$ sau elem.

```
sumaTemp = 0
for s in S:
    if sumaTemp + s <= k:
        sumaTemp += s.
    else if sumaTemp < max:   // suma temp > k, vrem. elem. max
        sumaTemp = max
```

LOAD BALANCE

La l-b avem optimul minim $\Rightarrow OPT \ge \frac{total}{2}$

Alg este α-aprox dacă $ALG \le \alpha \cdot OPT$

Avem încărcături de 80 și 120, n activități și e 1.1-aprox

a) timp cel mult /max 100?

$total = 80 + 120 = 200$

$OPT \ge \frac{200}{2} = 100$

$ALG = 120$

factor aprox $= \frac{120}{100} = 1.2 > 1.1$

Dacă cel mai lung task $\le 100 \Rightarrow OPT > 100$

ex: $100, 50, 50 \rightarrow total = 200$

$OPT \Rightarrow 100 + 50$ pe 1 mașină și 50 pe cealaltă $\Rightarrow OPT = 150$

Da e posibil ca factorul aprox $= 1.1$

Alt ex: $30, 40, 50, 80 \Rightarrow OPT$ avem $\begin{cases} 30+80 = 110 \\ 40+50 = 90 \end{cases}$ | $\Rightarrow ALG < 1.1 \cdot OPT$

ALG avem $\begin{cases} 30+50 = 80 \\ 40+80 = 120 \end{cases}$ | $120 < 121$

Tanislav Alexia
grupa 232

b) timp max pe activ = 10 ?

Dacă setul de activ se poate distribui egal la cele 2 mașini, atunci avem OPT = 100

$\frac{ALG}{100} = \frac{120}{100} = 1.2 > 1.1 \rightarrow$ nu are cum 1.1-apro

Nu sunt egal distribuite => dif. max. din cele 2 mașini pentru OPT va fi <= 10.

Dacă $S = 200$; a, b a.î. $\begin{cases} |a-b| <= 10 \\ a+b = 200 \\ a = max \text{, cu condițiile de sus} \end{cases}$ =>

=> $a = 105$ și $b = 95$ => $\frac{120}{105} = 1.14 > 1.1$ => tot nu are cum

② ALG_1 și ALG_2 => minimizare

ALG_1 este 2-aproximativ => $ALG_1(i) \leq 2 \cdot OPT(i)$ $\forall i$

ALG_2 este 4-aproximativ => $ALG_2(i) \leq 4\,OPT(i)$ $\forall i$

a) $\exists$ sigur un input i pt. care $ALG_2(i) \geq 2 \cdot ALG_1(i)$

Pp că $OPT(i) = 10$ -> $ALG_1(i) = 15$ pt că $15 \leq 2 \cdot 10$; $ALG_2(i) = 30$ pt că $30 \leq 4 \cdot 10$ => $ALG_2(i) = 30 = 2\,ALG_1(i)$

În cazul acesta -> prop. este adv.

b) $\nexists$ niciun i pt. care $2 \cdot ALG_2(i) \leq ALG_1(i)$

$ALG_1(i) \geq 2\,ALG_2(i)$ și $ALG_1(i) \leq 2\,OPT(i)$ => $2\,ALG_2(i) \leq 2\,OPT(i)$ =>

=> $ALG_2 \not\leq OPT$ (minimizare) dar $ALG_2(i) \leq 4\,OPT(i)$ => adv

TRAVELLING SALESMAN

① a) Pp ~~prin~~ că nu e NP-Hard

Avem $G(V,E)$ și construim $G'(V,E')$:

$\begin{cases} \text{pt. fiecare } e \in E' \text{ pt. care } e \in E \text{ avem } W(e, G') = 1 \\ \text{pt} \quad " \quad \text{— pt. care } e \in \bar{E} \text{ avem } W(e, G') = 2 \end{cases}$

(muchii $\in E$ au cost 1, iar $\notin E$ au cost 2)

Atunci G' = graf complet

G are ciclu hamiltonian ⇒ ∃ ciclu hamil. și în G' de cost n unde $n = |V|$ pt că toate muchiile vor avea ponderea 1

G nu are ciclu hamiltonian ⇒ soluția o să aibă cost minim $(n-1) + 2 = n+1$ ⇔ va fol. cel puțin o muchie din G'

Astfel, G' e obținut în timp polinomial (decidera ciclului hamiltonian este tot polinomială în alg. din ex)

Dar decidera cicl. hamil este NPC ⇒ este NP-Hard | ⇒ contradicție ⇒ + ipoteză

b) Dacă avem doar muchii de 1 și 2 :

$1, 1, 1 \Rightarrow 1+1 \geq 1$
$1, 1, 2 \Rightarrow 1+1 \geq 2$
$1, 2, 2 \Rightarrow 1+2 \geq 2$
$2, 2, 2 \Rightarrow 2+2 \geq 2$

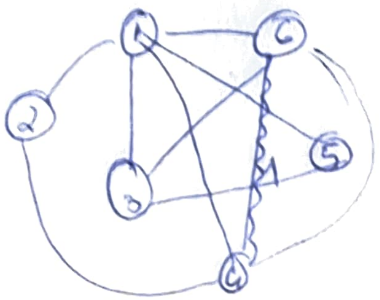
⇒ este satisfăcută inegalitatea triunghiului

c) Aplicăm alg. din curs pt. ponderile care nu respectă inegalitatea triunghiului.

Pp. că alg este 3/2 -aprox

Avem graful $\begin{cases} \text{toate muchiile cost 1} \\ \text{restul muchiilor care nu se află și sunt necesare pt a avea graf complet} \rightarrow \text{pondere 2} \end{cases}$

5/6



muchiile punctate nu sunt din MST

OPT : 1,2,4,6,3,5,1 => 6

MST: 1,2,1,3,1,4,1,5,1,6,1 => 10 <=> 1,2,3,5,3,6,1

$10/6 = 1.66 > 1.5$ => alg. ~~dacă~~ de la curs nu e $3/2$-aprox

...u Alexia
... 232

VERTEX COVER

a) worst case: $(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_4 \vee x_5) \wedge \ldots \wedge (x_1 \vee x_{m-1} \vee x_m)$

OPT = alegem x_1

ALG = ar putea alege mereu unul din cei 2 termeni rămași =>

și alege m termeni

factor = m-aprox unde m e nr. de clauze

b) În loc să alegem o singură variabilă din c_j, le luăm pe toate 3 și le setăm la True, iar în cel mai rău caz, alg. va alege de 3 ori mai multe variabile decât OPT.

c) v = vector = $[x_1, x_2, x_3 \ldots x_m]$

$x_i = \begin{cases} 1, & true \\ 0, & false \end{cases}$

minimizăm funcția $\sum_{i=1}^{m} x_i$

Constrângeri: $x_i + x_j + x_k \geq 1 \quad \forall c_t, t = \overline{1,m}$
$0 \leq x_i \leq 1, \quad \forall i \in \{1, m\}$

d) threshold = $\frac{1}{3}$ atunci $\begin{cases} x_i \leq \frac{1}{3} \Rightarrow x_i = 1 \\ x_i \geq \frac{1}{3} \Rightarrow x_i = 0 \end{cases}$

Dacă e cel mai rău caz => alegem toți x_i, $i \in \overline{1,m}$ și ne ducem la subpct. b) => 3-aprox

6/6