Gépköltséges ütemezési problémák

Gépköltséges ütemezés (Imreh, Noga 99)

Az új modellben a gépek száma nem a feladat paramétere, hanem az algoritmusnak kell megvásárolni a gépeket. Minden gép költsége 1.

A cél minimalizálni a gépvásárlás költségének és az ütemezés költségének (maximális befejezési idő) az összegét.

Az algoritmus

Egy növekvő ϱ sorozatra az A_ϱ algoritmus:
- minden kérés után a gépek i számát úgy határozzuk meg, hogy $\varrho_i \leq P < \varrho_{i+1}$ teljesüljön, ahol P az eddig érkezett munkák végrehajtási idejeinek összege
- Graham Lista algoritmusát használjuk a munkák ütemezésére.

tétel: Ha $\varrho = (0, 2, 6, \ldots, i(i+1), \ldots)$, akkor A_ϱ versenyképességi hányadosa $(1+\sqrt{5})/2$ a Lista modellben, ahol $\varrho = (0, 2, 6, \ldots, i(i+1), \ldots)$. Az algoritmus 1.693-versenyképes az idő modellben.

tétel: Nincs olyan algoritmus a Lista modellben, amely versenyképességi hányadosa kisebb $4/3$ -nál. Nincs olyan algoritmus a Lista modellben, amely versenyképességi hányadosa kisebb 1.186 - nál.

Bizonyítás

Vegyünk egy kis munkákból álló listát, legyen $p_i = \varepsilon$ minden i-re.
Könnyen látható, hogy ha az algoritmus nem vásárol második gépet, akkor nem konstans versenyképes.
Tehát feltehetjük, hogy az algoritmus a j_r munka után veszi a második gépét. Ha $P_j = \sum_{i=1}^{j} p_i \leq 2$, akkor az offline algoritmus csak egy gépet használ, így a költségek hányadosa legalább $\frac{P_r - \varepsilon + 2}{P_r + 1} \geq \frac{4 - \varepsilon}{3}$.
Ha $P_r > 2$, akkor egy offline algoritmus szét tudja osztani majdnem egyformán két gépen a munkákat, így a költségek hányadosa legalább $\frac{P_r - \varepsilon + 2}{P_r/2 + 2 + \varepsilon} \geq \frac{4}{3 + \varepsilon}$. Mivel ε tetszőlegesen kicsi lehet, ezért igazoltuk az állítást.

További eredmények

- Seiden 2000: véletlenített alsó korlátok
- He, Cai, 2002: félig-online probléma
- Dósa, He 2004: $1,5798$ - competitive algoritmus
- Yiang, He 2005: Preemptive online algoritmus
- Dósa, Tan 2010: $1,5486$ - versenyképes algoritmus, $\sqrt{2}$ alsó korlát

Visszautasításos és gépköltséges modell

Ebben a modellben a munkáknak van egy végrehajtási ideje és egy büntetése, ami annak a költsége, ha a munkát nem hajtjuk végre.

Továbbá a gépek száma nem egy adott paraméter, hanem az algoritmusnak kell megvásárolni őket.

A célfüggvény a gépek vásárlási költségének, a visszautasított munkákért fizetett büntetéseknek, és az ütemezés költségének az összege.

Visszautasításos és gépköltséges modell

A probléma általánosítása a síbérlési feladatnak, így adódik, hogy nincs olyan algoritmus, amely versenyképességi hányadosa kisebb, mint 2.

Kis munkák esetén van 2-versenyképes algoritmus.

A gépköltséges modellben adottak vásárlási szabályok, a visszautasításos problémára ismertek visszautasítási szabályok, kézenfekvő gondolat ezen szabályokat összeilleszteni.

Ez a megközelítés nem vezetett konstans versenyképes algoritmushoz.

Egy relaxált feladat

A visszautasításos feladatban a j munkának egy p_j végrehajtási ideje és egy w_j büntetése van. A relaxált feladatban használjuk a $T_A = M_A + P_A/M_A$ értéket munkáknak egy A halmazára, ahol, $P_A = \sum_{j \in A} p_j$ és $M_A = \max\{\sqrt{P_A}, \max_{j \in A} p_j\}$ ha $P_A > 1$ és $M_A = 1$ egyébként.

A relaxált feladatban azt az elfogadott halmazt kell megtalálnunk, amelyben a visszautasított munkák büntetése plusz az elfogadott munkák A halmazának T_A értéke minimális.

Egy J munkasorozatra jelölje J_k a J első k munkájából álló sorozatot.

RELOPT Algoritmus

Lemma Tegyük fel, hogy A^*_{k-1} egy optimális megoldása a relaxált problémának a J_{k-1} munkahalmaz esetén. Ekkor a J_k munkahalmazon vett relaxált feladatnak van olyan optimális megoldása, amelynek részhalmaza A^*_{k-1}.

A relaxált feladatnak egy a fenti lemmát kielégítő megoldássorozata megtalálható polinomiális időben. A megoldás az alábbi strukturális tulajdonságon alapul:

Lemma Minden j-re jelölje REL(j) azt a speciális változatát a relaxált problémának, ahol ki van kötve, hogy j-nél nagyobb végrehajtási idővel rendelkező munkát nem fogadhatunk el. Rendezzük a j-nél nagyobb munkákat a p_i/w_i érték szerint növekvő sorrendbe. Ekkor $REL(j)$-nek van olyan optimális megoldása, ami ezen sorozatnak egy prefixe.

OPTCOPY$_\varrho$ algoritmus

Legyen RELOPT az az online algoritmus, ami a fentieknek megfelelően megoldja a relaxált feladatot.

A j munka érkezésekor hajtsuk végre a következő lépéseket.

(i) Ha j-t *RELOPT* visszautasítja, utasítsuk vissza, egyébként (ii)

(ii) Ütemezzük a munkát az A_ϱ algoritmus alapján, ahol a gépvásárlás során csak az elfogadott munkákat vesszük figyelembe.

tétel OPTCOPY$_\varrho$ $(3+\sqrt{5})/2$-versenyképes, ahol $\varrho = (0, 4, \ldots, i^2, \ldots)$.

Általános gépköltségek

Ebben a modellben nem minden gép költsége 1, hanem egy $c(m)$ monoton növekvő költségfüggvény adja meg az első m gép megvásárlásának költségét.

Legyen $\varphi = \frac{1+\sqrt{5}}{2} \approx 1.618$ és $\mathcal{A} = A_\varrho$ ahol $\varrho = (0, c(2)\varphi, 2c(3)\varphi, \ldots, (i-1)c(i)\varphi, \ldots)$. Mivel c monoton növekvő, ezért ϱ egy növekvő sorozat.

tétel: A $\mathcal{A}$ algoritmus $1 + \varphi \approx 2.618$-versenyképes.

Bizonyítás

Legyen $\sigma = j_1, \ldots, j_n$ egy tetszőleges sorozat és rögzítsünk egy optimális megoldást. Jelölje m az $\mathcal{A}$ által vett gépek számát, legyen j_r az a munka, amelyet utoljára fejez be, és k legyen $\mathcal{A}$ gépeinek száma j_r ütemezésekor.

Mivel $\mathcal{A}$ mindig a legkisebb töltést tartalmazó géphez rendeli a munkát, ezért

$$\mathcal{A}(\sigma) \leq c(m) + \frac{P_r - p_r}{k} + p_r.$$

I. eset $k < m$.

Ekkor $P_r < kc(k+1)\varphi$ ezért $P_r/k \leq c(k+1)\varphi \leq c(m)\varphi$.

Ha az optimális megoldás legalább m gépet használ, akkor a gépvásárlás költsége legalább $c(m)$, az ütemezésé legalább p_r így
$\mathcal{A}(\sigma) \leq c(m) + \frac{P_r - p_r}{k} + p_r \leq (1+\varphi)c(m) + p_r \leq (1+\varphi)opt(\sigma)$.

Ha az optimális megoldás m-nél kevesebb gépet használ, akkor az ütemezési költség legalább $c(m)\varphi$, mivel a végrehajtási idők összege legalább $(m-1)c(m)\varphi$.

Tehát $c(m) \leq \mathrm{opt}(\sigma)/\varphi$. Másrészt $P_r/k \leq c(k+1)\varphi \leq c(m)\varphi \leq \mathrm{opt}(\sigma)$ és $p_r \leq \mathrm{opt}(\sigma)$, tehát $\mathcal{A}(\sigma) \leq (2 + 1/\varphi)\mathrm{opt}(\sigma) = (1+\varphi)\mathrm{opt}(\sigma)$.

II. eset $k = m$.

Ha $k = m$, akkor $\mathcal{A}(\sigma) \leq c(m) + \frac{P_r - p_r}{m} + p_r$.

Ha OPT m-nél kevesebb gépet használ, akkor az ütemezési költsége legalább $P_r/(m-1) \geq c(m)\varphi$. Ebből adódik, hogy $c(m) \leq \mathrm{opt}(\sigma)/\varphi$. Másrészt $P_r/m \leq \mathrm{opt}(\sigma)$ és $p_r \leq \mathrm{opt}(\sigma)$, így $\mathcal{A}(\sigma) \leq (2 + 1/\varphi)\mathrm{opt}(\sigma) = (1+\varphi)\mathrm{opt}(\sigma)$.

Ha OPT m gépet használ, akkor a vásárlás költsége $c(m)$, az ütemezésé $\max\{p_r, P_r/m\}$. Tehát $\mathcal{A}(\sigma) \leq 2\mathrm{opt}(\sigma)$.

Végül, ha OPT m-nél több gépet használ, akkor vásárlási költsége $c(m+1) \geq c(m)$ az ütemezésé legalább p_r. Másrészt $P \leq mc(m+1)\varphi$ így $P_r/m \leq c(m+1)\varphi \leq \varphi\mathrm{opt}(\sigma)$. Tehát $\mathcal{A}(\sigma) \leq (1+\varphi)\mathrm{opt}(\sigma)$.

Kis munkák esete

Tegyük fel, hogy a munkák méretét korlátozza a legolcsóbb gép költsége, azaz $p_i \leq c(m) - c(m-1)$ teljesül minden $i \geq 1$ és $m \geq 1$ esetén (a $c(0) = 0$ értéket használva).

Legyen $\mathcal{B} = \mathcal{A}_\varrho$ a $\varrho = (0, c(2), 2c(3), \dots, (i-1)c(i), \dots)$ sorozatra.

tétel $\mathcal{B}$ versenyképességi hányadosa 2 a kis munkák esetén.

tétel Nincs olyan online algoritmus, amelynek 2-nél kisebb a versenyképességi hányadosa. Az alsó korlát igaz kis munkák esetén is.

Összefüggő gépek esete

Ebben a modellben két halmazból vásárolhatunk gépeket S_1 1 sebességű gépeket tartalmaz, S_2 pedig $s > 1$ sebességű gépeket.

Az s sebességű gép s-szer gyorsabban hajtja végre a munkákat.

A c_1 nemcsökkenő függvény adja meg az S_1-beli munkák vásárlási költségét, c_2 pedig az S_2 beli munkákét.

Mohó algoritmus

Az algoritmus használja az OPT_r értéket, ami az optimális offline megoldás célfüggvényértéke az első r munkára.

Amikor j_r megérkezik, akkor a Mohó algoritmus annyi gépet vesz, hogy az egyes halmazokra a hozzájuk tartozó gépek i_1 és i_2 számára $c_1(i_1) \leq OPT_r < c_1(i_1 + 1)$ és $c_2(i_2) \leq OPT_r < c_2(i_2 + 1)$ teljesül.

Ezt követően az algoritmus a j_r munkát a Lista szabály szerint ütemezi, ahhoz a géphez rendeli, ahol a töltés minimális lesz a hozzárendelés után.

Eredmények

tétel Mohó versenyképességi hányadosa 6.

Vizsgáljuk a rögzített gépszám esetét, azaz azt a speciális esetet, amikor n gép adott 1 sebességgel és m gép adott s sebességgel. ($c_1(k) = 0$ ha $k \leq n$, $c_1(k) = \infty$ ha $k > n$, $c_2(k) = 0$ ha $k \leq m$, $c_2(k) = \infty$ ha $k > n$.)

tétel A rögzített gépszám esetén Mohó versenyképességi hányadosa 4.

Megjegyzés Az MG algoritmus (Imreh 2003) versenyképességi hányadosa 3 rögzített gépszám esetén.

MG algoritmus

- Ha $ms \geq n$ akkor minden munkát ütemezzünk az S_2 géphalmazon a LISTA ütemezési algoritmus alapján.
- ha $ms < n$ akkor a következő algoritmust használjuk:
  - 1. Legyen $R := \emptyset$.
  - 2. A j munka érkezésekor, legyen r egy alsó korlát a $R \cup \{j\}$ munkáknak az S_2 gépen való ütemezési költségére. Ha $p_j \geq r$, akkor
    - (a) Rendeljük j-t S_2-höz,
    - (b) Legyen $R = R \cup \{j\}$.
  - 3. Egyébként rendeljük j-t S_1-hez.
  - 4. Ütemezzük a munkát a Lista algoritmus szerint azon a géphalmazon, amelyhez hozzá lett rendelve.

Folytonos eset

Téglalapokat szeretnénk átfedés és forgatás nélkül egy bennfoglaló téglalapba pakolni, a kerületet minimalizálva.

Az online sávpakolásnak is egy kiterjesztett változata, ott az egyik dimenzió korlátozott.

A feladat megoldására sávpakolási algoritmusok használhatóak.

Az NFS algoritmus

Minden j-re legfeljebb egy 2^j magas polc van.

Egy $p_i = (w_i, h_i)$ elemhez kiválasztjuk azt a k-t, amire $2^{k-1} < h_i \leq 2^k$.

Ha van aktív polc, ami 2^k magas, akkor rakjuk a polcra annyira balra, amennyire lehet. Ha a szélesség elérte az aktuális magasság -szorosát,a kkor zárjuk el a polcot.

Ha nincs aktív polc, vagy nem tudjuk az elemet az aktív polcra rakni, akkor definiálunk egy új 2^k magas polcot és az lesz az aktív polc, és arra rakjuk az elemet. Ha a szélesség elérte az aktuális magasság -szorosát, akkor zárjuk el a polcot.

Tétel Az algoritmus $(4 + \sqrt{2/\alpha})(1 + \alpha)$-versenyképes.

Csökkenő magasságú téglalapok

SDH algoritmus

1. Rakjuk a tárgyat az aktuális polcra annyira a balra, amennyire lehetséges, ha a polc szélessége legfeljebb az aktuális konténer magassága.
2. Ellenkező esetben zárjuk le az aktuális polcot, és nyissunk egy új aktuális polcot az eddigi polcok tetején, majd rakjuk annak bal sarkába a tárgyat.

Elemzés

Tétel: SHD $C \leq \sqrt{3} + 1 \approx 2.7321$- versenyképes.

Alapötlet: Legyenek $h_1, \ldots, h_m$ a polcok magasságai. Továbbá H a konténer magassága és w_{max} a maximális szélessége az elemeknek.

Ekkor az algoritmus költsége legfeljebb $2H + w_{max}$.

Másrészt az i-dik polcon a lefedett terület legalább $(h_1 + \cdots + h_i)h_{i+1}$.
Tehát a teljes terület legalább $(H^2 - h_1 H)/2$, amiből azt kapjuk, hogy $OPT \geq 2\sqrt{H^2 - h_1 H)/2}$.

Szintén $OPT \geq h_1 + w_{max}$.

Rugalmas elemek pakolása

Ebben a modellben az elemek mérete nem adott, hanem változtatható a terület fixen hagyása mellett.

Az offline optimum egy négyzetet hoz létre, költsége $\sqrt{T}$, ahol T a tárgyak összterülete.

Legyen T(k) a k-dik elem területe, $a(k), b(k)$ a módosított oldalak. Továbbá $A(k) \leq B(k)$ jelöljék a konténer oldalait a k-dik elem pakolása előtt.

Expand Algoritmus

Az első elemet négyzetté alakítjuk, majd berakjuk a tengelyekkel párhuzamosan. A többieket így pakoljuk:

1. Eset Ha az elem kicsi ($T(k) \leq B(k)^2$), akkor $b(k) = B(k)$ és $a(k) = T(k)/b(k) \leq B(k)$, majd a konténer hosszabb oldalához ragasztjuk az elemet és
$B(k+1) = \max\{B(k), A(k) + a(k)\}$,
$A(k+1) = min\{B(k), A(k) + a(k)\}$.
2. Eset Ha az elem nagy ($T(k) > B(k)^2$), akkor $a(k) = b(k) = \sqrt{T(k)}$, és az elemet a konténer nagyobb odlalához ragasztjuk: $A(k+1) = a(k)$ és
$B(k+1) = A(k) + a(k)$. (Egy lyuk keletkezik, amit nem használunk később.)