Versenyképesség általánosításai

Extra erő az online algoritmusnak

- előrenéző tulajdonság: az algoritmus az input valamely részét látja
- Erőforrás kiterjesztés: az offline algoritmus kisebb mennyiségű erőforrást használhat (példát láttunk a weblapletöltési problémánál.
- az algoritmus több megoldást építhet és ezekből a jobbat vesszük figyelembe.
- apró módosításokat hajthat végre a régebbi döntéseken (pl ládapakolásnál átpakolhat)

Megszorítás az ellenfélnek

- Rendezett input: az input nem lehet tetszőleges, hanem valamely szabály alapján rendezve van.
- Igénylési gráf: a lapozási probléma esetén olyan modell, amelyben az input nem lehet tetszőleges sorozat, hanem egy gráf (program gráfja) pontjait kapjuk, minden pont után csak egy szomszédja jöhet.
- Félig átlagos elemzés: az ellenfél generálhatja az input sorozat tartalmát, de az input a definiált sorozat egy véletlen permutációja lesz (egyenletes eloszlás alapján).
- Alkalmazkodó függvény. Olyan problémáknál használható, amelyben adott paraméter valami erőforrás (pl. gépek száma). Ekkor egy input α sorozat, ha $OPT_{\alpha n} = OPT_{n'}$ minden $n' \geq \alpha n$ esetén, azaz αn mennyiség fölé növelve az erőforrások mennyiségét az optimális megoldás célfüggvényértéke nem változik. Az alkalmazkodó függvény minden α esetén α sorozatok mellett vizsgálja a versenyképességi hányadost.

Előrenéző algoritmusok (Lapozás)

Erős előrenézés: (Albers 93)
Az algoritmus azt a legrövidebb prefixet látja, amely l különböző lapra vonatkozó kérést tartalmaz.

LRU(l) algoritmus Az előre látott szakaszban nem igényelt lapok közül a legrégebben használtat dobjuk ki.

Tétel: Az LRU(l) algoritmus versenyképességi hányadosa $k-l$, ha $l \leq k-2$.

Tétel Nincs olyan erősen k-előrenéző online algoritmus, amelynek versenyképességi hányadosa kisebb, mint $k-l$, ha $l \leq k-2$.

Természetes előrenézés: (Breslauer 96) Az algoritmus a leghosszabb sorozatot látja, amely l különböző lapra vonatkozó kérést tartalmaz.

Tétel: Az LRU(l) algoritmus versenyképességi hányadosa $(k+l)/(l+1)$.

Tétel Nincs olyan természetesen k-előrenéző online algoritmus, amelynek versenyképességi hányadosa kisebb, mint $(k+l)/(l+1)$.

Igénylési gráf (lapozás)

Az igénylési gráf $G = (V,E)$:

- a csúcsok a lapok
- két csúcsot akkor köt össze él, ha előfordulhatnak egymás után a kérések listájában.

Tehát az input egy séta az igénylési gráfon.

- $c_{A,k}(G)$ jelöli a G gráfon a versenyképességi hányadost.
- $c_k(G)$ a lehető legjobb hányados a gráfon.

FAR egy bélyegző algoritmus, amely mindig azt a jelöletlen lapot rakja ki, amelynek a kért laptól a távolsága maximális.

Tétel: Minden G és k esetén
$c_{\mathrm{FAR},k}(G) = O(c_k(G))$.

Tétel: LRU versenyképessége egyetlen gráf esetén sem nagyobb, mint FIFO versenyképessége.

Félig átlagos elemzés

Az ellenfél generálhatja az input sorozat tartalmát, de az input a definiált sorozat egy véletlen permutációja lesz (általában egyenletes eloszlás alapján).

Online kiszolgáló elhelyezés

Adott metrikus tér és benne kérések egy U halmaza. A cél kiszolgálók egy olyan F halmazát megtalálni, amelyre minimális

$f|F| + \sum_{u \in U} min_{f \in F} d(u, f)$

Az online feladatban a kérések egyenként jönnek, és vagy egy meglevő kiszolgálóhoz kell hozzárendelni őket vagy újat nyitni.

R Algoritmus: (Meyerson 2001) Ha a kérés távolsága δ a legközelebbi kiszolgálótól, akkor nyissunk abban a pontban egy új kiszolgálót δ/f valószínűséggel.

Tétel: Az algoritmus $O(\log n)$ versenyképes.

Tétel: Nincs konstans versenyképes algoritmus.

Félig átlagos eset: Az ellenfél megadhatja a kérések halmazát, de a sorrend egy egyenletes eloszlás alapján választott permutáció.

Tétel: Az algoritmus konstans versenyképes a félig átlagos esetre.

Elmozdítható kiszolgálók

Vannak alkalmazások, ahol nem indokolt, hogy a telepített kiszolgáló helye ne legyen változtatható. A változtatás költségétől függően 2 modellt definiálunk:

- ingyenes mozgatás
- konstans költségű mozgatás

Ingyenes korlátnál legyen két-két kérés a 0 és r pontokba. Amennyiben az első négy kérésnél két kiszolgálót vettünk, jön további két kérés az r/2 pontba. Az optimális megoldás 1 kiszolgálót használ a költsége $2r+1$, az online algoritmus legalább két kiszolgálót használ, a költsége legalább $2+r$. A versenyképesség alsó korlátja max(1+2r)/2,(2+r)/(1+2r)

2-versenyképes algoritmus: Az adott igény érkezése után határozzuk meg az optimális offline megoldást. Ha a kiszolgálok száma nagyobb, mint az algoritmus által eddig vásárolt kiszolgálók száma, növeljük a kiszolgálók számát, és mozgassuk a kiszolgálókat az optimális helyekre. Egyébként oldjuk meg rögzített kiszolgálószám mellett a minimális költségű kiszolgálást

Rendezett inputok

Ekkor az input nem lehet tetszőleges, hanem valamely szabály alapján rendezve van. Ilyen esetekben az online algoritmusok gyakran jobb versenyképességgel rendekeznek.

Ládapakolás csökkenő méretű elemekkel

Tétel: Next Fit aszimptotikusan 1.691-versenyképes (2 helyett)

Tétel: First Fit aszimptotikusan 1.222-versenyképes (1.7 helyett)

Tétel: Nincs $8/7$-nél kisebb versenyképességi hányadossal rendelkező online ládapakolási algoritmus csökkenő méretű elemekre.

Ütemezés

Tétel: Lista $4/3 - 1/(3m)$-versenyképes csökkenő méretű munkák esetén ($2 - 1/m$ helyett).

Bizonyítás: Az állítást indirekt igazoljuk. Ehhez tegyük fel, hogy léteznek olyan ellenpéldák, amelyekre az optimális ütemezés és az LPT algoritmus által kapott ütemezés költségeinek hányadosa nagyobb mint $4/3 - 1/(3m)$. Ezen ellenpéldák közül van olyan, amely minimális számú munkát tartalmaz.

Mivel a tekintett ellenpéldában a munkák száma minimális, ezért az utolsónak ütemezett munka befejezési ideje megegyezik a maximális befejezési idővel. Amennyiben ez a tulajdonság nem teljesülne, akkor arra a σ' inputra, amelyet a tekintett ellenpéldából az utolsó munkát elhagyva kapnánk $LPT(\sigma') = LPT(\sigma)$ és $OPT(\sigma) \geq OPT(\sigma')$ teljesülne.

A fentiek alapján azt kapjuk, hogy az utolsó munka kezdési ideje $LPT(\sigma) - p_n$. Másrészt eddig az időpontig az összes gép dolgozott, így

$$LPT(\sigma) - p_n \leq \frac{\sum_{i=1}^{n-1} p_i}{m}.$$

Másrészt $OPT(\sigma) \geq \frac{1}{m}(\sum_{j=1}^{n} p_j)$.
Következésképp

$$\frac{4}{3} - \frac{1}{3m} < \frac{LPT(\sigma)}{OPT(\sigma)} \leq \frac{p_n + \sum_{i=1}^{n-1} p_i/m}{OPT(\sigma)} =$$

$$\frac{p_n(1-1/m)}{OPT(\sigma)} + \frac{\sum_{i=1}^{n} p_i/m}{OPT(\sigma)} \leq \frac{p_n(1-1/m)}{OPT(\sigma)} + 1.$$

A fenti egyenlőtlenség jobb és baloldalát véve a következő korlát adódik $OPT(\sigma)$ értékére:

$$OPT(\sigma) < 3p_n.$$

Mivel p_n a minimális végrehajtási idő, ezért a fenti egyenlőtlenség azt jelenti, hogy az optimális ütemezésben minden gép legfeljebb két munkát tartalmaz, azaz $n \leq 2m$. Másrészt ebben az esetben az LPT eljárás optimális, azaz ellentmondáshoz jutottunk, amivel a tétel állítását igazoltuk.

Egyéb online modellek

- online klaszterezési eljárások
- online gráfszínezés
- online navigáció
- online portfolió választás