Online algoritmusok kombinatorikai problémákra

Doktori (Ph.D.) értekezés tézisei

Nagy-György Judit

Témavezető: Hajnal Péter
Egyetemi docens

Matematika- és Számítástudományok Doktori Iskola
Szegedi Tudományegyetem
Bolyai Intézet

2009

1. Bevezetés: kompetitív elemzés

Az online algoritmusokat körülbelül 30 éve vizsgálják. Hatékonyságuk mérésének egyik fő módszere egyfajta ,,legrosszabb eset elemzés", az úgynevezett kompetitív analízis. Egy online problémában az input részletenként érkezik. Az inputsorozaton adott egy rendezés, a sorozat tagjai eszerint a rendezés szerint egyesével jönnek, és az *online algoritmus*nak olyan döntések sorozatát kell hoznia velük kapcsolatban, amelyeknek hatása lesz a teljes hatékonyság végső minőségére. Mindezen döntéseket a már megérkezett inputrész alapján kell hozni anélkül, hogy bármilyen információja lenne a jövőre vonatkozóan.

Minden I inputhoz lehetséges outputok egy halmaza tartozik, és minden lehetséges outputhoz hozzá van rendelve egy pozitív valós érték, amely az output költsége I inputon. Ezek közül a minimálisat $\mathrm{opt}(I)$-vel jelöljük. Az $\mathcal{A}$ algoritmus minden I inputra kiszámít egy lehetséges outputot. Az ehhez tartozó költséget $\mathcal{A}(I)$-vel jelöljük.

Egy online algoritmus *c-kompetitív (szigorú értelemben)* valamely $c > 0$-ra, ha minden véges inputsorozatra

$$\mathcal{A}(I) \leq c \cdot \mathrm{opt}(I).$$

$\mathcal{A}$ versenyképességi hányadosa a legkisebb olyan c, amelyre $\mathcal{A}$ c-kompetitív. Egy $\mathcal{A}$ algoritmus *gyengén c-kompetitív*, ha van olyan a konstans, hogy minden véges I inputsorozat esetén

$$\mathcal{A}(I) \leq c \cdot \mathrm{opt}(I) + a.$$

$\mathcal{A}$ gyenge versenyképességi hányadosa a legkisebb olyan c, amelyre $\mathcal{A}$ gyengén c-kompetitív.

2. Visszautasításos gépköltséges ütemezés

Az ütemezési problémáknak nagy irodalma van, és több modellje ismert (ld. [18]). Az egyik legalapvetőbb és legegyszerűbb a lista modell, amelyben rögzített számú gép adott és a munkák egy listából érkeznek. Minden munkának van *végrehajtási ideje*. A ,,párhuzamos gépek esetében" adott m gép. Egy munka beütemezésén valamely géphez való hozzárendelését értjük. Minden munkát be kell ütemezni valamely gépre és egy gép nem futtathat két munkát párhuzamosan. Egy gép futási ideje a hozzá rendelt munkákhoz tartozó végrehajtási idők összege. A költség a futási idők maximuma, ennélfogva a cél ennek minimalizálása.

A probléma online változatában a munkák (és a hozzájuk tartozó végrehajtási idők) egyesével jelennek meg. Amikor egy munka megérkezik, hozzá kell rendelni egy géphez anélkül, hogy bármilyen információnk lenne a jövőbeli munkákról. A Graham [7] által 1966-ban fejlesztett $\mathcal{LIST}$ algoritmus az első ebben a modellben. Amikor a j-edik munka megjelenik, a $\mathcal{LIST}$ algoritmus ahhoz a géphez rendeli, amelynek az aktuális futási ideje minimális. Graham [7] bebizonyította, hogy a $\mathcal{LIST}$ algoritmus versenyképességi hányadosa $2 - 1/m$.

A gépköltséges ütemezési problémát [8]-ben definiálták. Ebben a modellben a gépek száma nem a probléma adott paramétere: az algoritmusnak meg kell vásárolnia a gépeket, és a cél a

gépek vásárlására fordított költség plusz a futási idők maximumának minimalizálása. [8]-ben vizsgálták azt a problémát, amelyben minden gép vásárlási költsége 1. A munkák egyesével érkeznek és a döntéshozónak minden lépésben döntenie kell az új gépek vásárlásáról majd be kell ütemezni a legutóbb érkezett munkát egy már megvásárolt gépre, mindezt anélkül, hogy bármilyen információval rendelkezne a jövőbeli munkákkal kapcsolatban. Imreh és Noga [8] megkonstruálták az $\mathcal{A}_\rho$ algoritmust, ahol $\rho = (0 = \rho_1, \rho_2, \dots, \rho_i, \dots)$ egy növekvő sorozat. Bebizonyították, hogy az $\mathcal{A}_\rho$ algoritmus versenyképességi hányadosa $\varphi = (1 + \sqrt{5})/2$ a $\rho = (0, 4, 9, 16, \dots, i^2, \dots)$ sorozatra.

A visszautasításos ütemezési problémát [3]-ban definiálták. Ebben a modellben lehetőség van a munkák elutasítására. A munkákat a végrehajtási idejük és a hozzájuk tartozó büntetés jellemzi. A költség a futási idők maximuma plusz az elutasított munkák büntetése ebben a problémában. Bartal és mtsai [3] algoritmusa a Reject-Total-Penalty(α). Ennek α paramétere küszöb szerepet játszik. A szerzők bebizonyították, hogy az $\mathcal{RTP}(\varphi - 1)$ algoritmus $(1 + \varphi)$-kompetitív.

A fejezet következő eredményei [14]-ben találhatóak.

Itt az általánosabb MCR modellt tekintetük, amely a fenti két megközelítés kombinációja. Ebben a gépek nem adottak előre, de az algoritmus megvásárolhatja azokat, ezenkívül a munkákat el is lehet utasítani. A költség a gépek vásárlására fordított költség plusz a futási idők maximuma plusz az elutasított munkák büntetése, ezt az összeget kell minimalizálni. Feltesszük, hogy a gépek ára 1.

A problémábann a j-edik munkának p_j a végrehajtási ideje, és w_j az elutasításával járó büntetés. $P_H = \sum_{j \in H} p_j$ és $W_H = \sum_{j \in H} w_j$.

Mivel a gépvásárlásos és az elutasításos modellt ötvöztük, természetes ötlet kombinálni a két eredeti modell algroitmusait úgy, hogy az összetett problémát oldja meg. Az első részben megmutattuk azt a meglepő eredményt, hogy az az említett algoritmusok egyszerű kombinációinak nincs konstans versenyképességi hányadosa.

Ezután adtunk egy szofisztikáltabb algoritmust. Az alapötlet a relaxált változat vizsgálata az eredeti probléma helyett, amelyben az ütemezés költségét (gépek ára plusz a futási idők maximuma) helyettesítjük annak egy alsó korlátjával.

Tegyük fel, hogy elfogadtuk munkák egy halmazát, jelölje indexeik halmazát A, továbbá m gépet vásároltunk, és az aktuális futási idők maximuma M. Ekkor az ütemezés költségére teljesül az $M + m \geq M + P_A/M$ összefüggés. Definiáltuk a következő kifejezést:

$$M_A := \begin{cases} \max\{\sqrt{P_A}, l_A\}, & \text{if } P_A > 1 \\ 1 & \text{otherwise} \end{cases}$$

Indexek egy tetszőleges A halmazára

$$T_A := \begin{cases} M_A + \dfrac{P_A}{M_A} & \text{if } A \neq \emptyset \\ 0 & \text{if } A = \emptyset \end{cases}$$

Definiáltuk a *relaxált problémát*: munkák érkeznek, mindegyikhez tartozik végrehajtási idő és büntetés. Olyan megoldást kell találni, amelyre az elutasított munkák összbüntetése és az elfogadott munkákhoz tartozó T_A érték összege minimális. Indexek egy J halmazára a relaxált probléma optimális megoldásának költségét jelölje $\mathrm{ropt}(J)$. a következő állítások igazak.

13. következmény *Munkák indexeinek tetszőleges J halmazára* $\mathrm{ropt}(J) \leq \mathrm{opt}(J)$.

Ahhoz, hogy az Optcopy algoritmust megkonstruáljuk, a relaxált probléma optimális megoldásának struktúráját kell megvizsgálnunk. Indexek egy J halmaza esetén J_k jelölje J az első k indexének halmazát. Ekkor a következő állítás teljesül.

14. lemma *Tegyük fel, hogy a J_{k-1} halmazon a relaxált probléma egy optimális megoldáshoz tartozó elfogadott halmaz A^*_{k-1}. Ekkor a J_k halmazon a relaxált problémának van olyan megoldása, amelyre A^*_{k-1} részhalmaza az elfogadott munkákhoz tartozó indexek halmazának.*

A relaxált probléma polinom időben megoldható.A problémát megoldó algoritmus a következő strukturális tulajdonságon alapul.

15. lemma *A j-edik munkához tekintsük a* $\mathrm{REL}(j)$ *problémát, amely a megszorított relaxált probléma, amelyben j a legnagyobb elfogadott munka indexe. Rendezzük a j-nél nem nagyobb indexű munkákat p_i/w_i szerinti növekvő sorba. Ekkor* $\mathrm{REL}(j)$*-nek van olyan optimális megoldása, amely egy kezdőszelete ennek a sorozatnak.*

A 15. lemma alapján polinom idejű algoritmus, amely olyan optimális megoldást ad a relaxált problémákra, amelyek a 14. lemmát is kielégítik. Ezt az algoritmust *Relopt*nak nevezzük. Jelölje a J_k-ból elfogadott munkák indexeinek halmazát A^*_k és az elutasított munkák indexeinek halmazát R^*_k. Ennélfogva $A^*_i \subseteq A^*_k$ ha $i \leq k$. a következő állítások igazak.

16. lemma *A fent definiált halmazokra a következő egyenlőtlenségek teljesülnek:*

$$\sum_{j=1}^{n} W_{R^*_{j-1} \cap A^*_j} \;\leq\; T_{A^*_n}.$$

Definiáltuk az $\mathrm{Optcopy}_\rho$ algoritmusok halmazát.

$\mathcal{OC}_\rho$ algoritmus

Új munka érkezésekor hajtsuk végre a következő lépéseket.

(i) Ha Relopt elutasítja, utasítsuk el, különben menjünk a (ii) lépésre

(ii) Ütemezzük a munkát $\mathcal{A}_\rho$ szerint, ahol a gépvásárlási szabályokat csak az elfogadott munkák alapján alkalmazzuk.

---

17. tétel *Az $\mathcal{OC}_\rho$ algoritmus a $\rho = (0, 4, 9, 16, \ldots, i^2, \ldots)$ sorozattal $(\varphi + 1)$-kompetitív.*

3. Gráfok és hipergráfok színezése

Az online gráf fogalma [11]-ban jelent meg, majd az online hipergráfot, amely az online gráf általánosítása, [1]-ben definiálták először. Az *online hipergráf* egy $H^{\prec} = (H, \prec)$ struktúra, ahol $H = (V, E)$ egy hipergráf és $\prec$ a csúcsok egy rendezése. H_i jelöli a $\prec$ szerinti első i csúcsból álló V_i halmaz által feszített hipergráfot. Egy online hipergráfszínező algoritmus az i-edik csúcsot kizárólag a H_i részhipergráf ismeretében színezi. Egy $\mathcal{A}$ online algoritmus és $H^{\prec}$ online hipergráf esetén a költség az $\mathcal{A}$ algoritmus által használt színek száma, amelyet $\chi_{\mathcal{A}}(H^{\prec})$-val jelölünk. Nyilván, $\mathrm{opt}(H^{\prec}) = \chi(H)$. Egy H hipergráf esetén $\chi_{\mathcal{A}}(H)$ jelöli a $\chi_{\mathcal{A}}(H^{\prec})$ értékek maximumát az összes $\prec$ rendezésen. Az online gráfszínezést több cikkben vizsgálták, részletesen erről a [9] összefoglaló munkában olvashatunk.

3.1. Tiltott részgráfokra vonatkozó eredmények

Az alfejezet eredményei [13]-ben találhatóak.

A G gráf $g(G)$ derékbősége a legrövidebb körének hossza, a $g_o(G)$ pedig a páratlan derékbősége, amely a legrövidebb páratlan körének hossza. Az u és v csúcsok $\mathrm{dist}(u, v)$ távolsága a legrövidebb uv út hossza. Egy d pozitív egészre $N_d(v)$ a v-től legfeljebb d pozitív távolságra lévő csúcsok halmaza. $N_{d,\mathrm{odd}}(v)$ azon csúcsok halmaza, amelyek távolsága v-től legfeljebb d páratlan. Minden $S \subset V(H)$-ra definiáljuk $N_d(S) = \bigcup_{v \in S} N_d(v) \setminus S$ és $N_{d,\mathrm{odd}}(S) = \bigcup_{v \in S} N_{d,\mathrm{odd}}(v) \setminus S$ halmazokat. Legyen $N_d^{\prec}(v)$ azon v-t megelőző csúcsok halmaza, amelyek távolsága v-től legfeljebb d pozitív.

A [10]-ben ismertetett $\mathcal{B}_n$ algoritmus kevesebb mint $2n^{1/2}$ színnel színezi az n csúcsú C_3- és C_5-mentes gráfokat. Ezt az algoritmust általánosítottuk nagy derékbőségű gráfokra.

$\mathcal{B}_{n,d}$ algoritmus

Tekintsük a $G^{\prec}$ gráf $v_1 \prec \ldots \prec v_n$ inputsorozatát, amelyre $g(G) > 4d + 1$. Inicializáljuk $U_i = \emptyset$ halmazokat minden $i > dn^{1/(d+1)}$-re.
s-edik lépés:

(i) Ha van $i \in [dn^{1/(d+1)}]$, hogy v_s nem szomszédos egyik i színű csúccsal, akkor színezzük v_s-t a legkisebb ilyen i-vel.

(ii) Különben ha van olyan $i > dn^{1/(d+1)}$, hogy $v_s \in N_d(U_i)$, akkor színezzük v_s-t a legkisebb ilyen i-vel.

(iii) Különben legyen j a legkisebb $i > dn^{1/(d+1)}$ egész, amelyre $U_i = \emptyset$. Állítsuk be $U_j = N_d^{\prec}(v_s)$-t és színezzük v_s-t j-vel.

---

21. tétel *A $\mathcal{B}_{n,d}$ algoritmus az n csúcsú $g > 4d+1$ derékbőségű $G^{\prec}$ gráfot jólszínezi legfeljebb $(d+1)n^{1/(d+1)}$ színnel.*

A második általánosítás $\mathcal{BO}_{n,d}$, amely Lovász n' csúcsú páros gráfokat legfeljebb $\log_2 n'$ színnel színező $\mathcal{AA}$ algoritmusát (ld. [9]) használja segédalgoritmusként.

$\mathcal{BO}_{n,d}$ algoritmus

Tekintsük a $G^{\prec}$ gráf $v_1 \prec \ldots \prec v_n$ inputsorozatát, amelyre $g_o(G) > 4d+1$. Legyen $r = (n/(2d\log_2 2d))^{1/2}$. Inicializáljuk $S_i = \emptyset$ halmazokat minden $i \in [r]$-re és $U_i = \emptyset$ halmazokat minden $i > r$-re.
s-edik lépés:

(i) Ha van olyan $i \in [r]$, hogy az $S_i \cup \{v_s\}$ által feszített részgráf 2-színezhető és $\mathcal{AA}$ legfeljebb $2\log_2 2d$ színt használ $S_i \cup \{v_s\}$ kiszínezésére, akkor legyen j a legkisebb ilyen i. Legyen $S_j = S_j \cup \{v_s\}$-t és színezzük v_s-t (az S_j által feszített részgráfban) az $\mathcal{AA}$ algoritmus szerint a $2(j-1)\log_2 2d + 1, \ldots, 2j\log_2 2d$ színeket használva.

(ii) Különben ha van olyan $i > r$, hogy $v_s \in N_{d,\mathrm{odd}}(U_i)$, akkor színezzük v_s-t a legkisebb ilyen i-vel.

(iii) Különben legyen j a legkisebb $i > r$ egész, amelyre $U_i = \emptyset$. Állítsuk be $U_j = N^{\prec}_{d,\mathrm{odd}}(v_s) \cap (\bigcup_{\ell=1}^{r} S_\ell)$-t és színezzük v_s-t j-vel.

23. tétel *A $\mathcal{BO}_{n,d}$ algoritmus az n csúcsú $g > 4d+1$ páratlan derékbőségű $G^{\prec}$ gráfot jólszínezi legfeljebb $2(2n\log_2 2d/d)^{1/2}$ színnel.*

3.2. Eredmények hipergráfokon

Az alfejezet eredményei [15]-ben találhatóak.

Ebben a részben 2-színezhető k-uniform hipergráfokat tekintettünk $k \geq 3$-ra Egy természetes online hipergráfszínező algoritmus a First Fit ($\mathcal{FF}$). Amikor egy csúcs érkezik, $\mathcal{FF}$ a legkisebb olyan színt rendeli hozzá, amely nem eredményez egyszínű élet.
A következőket bizonyítottuk.

24. tétel *Minden $\mathcal{A}$ online hipergráfszínező algoritmushoz létezik n csúcsú 2-színezhető k-uniform H hipergráf, amelyre $\chi_{\mathcal{A}}(H) \geq \lceil n/(k-1)\rceil$. Ha H egy n csúcsú k-uniform hipergráf, akkor $\chi_{\mathcal{FF}}(H) \leq \lceil n/(k-1)\rceil$.*

A tétel azt mutatja, hogy $\mathcal{FF}$ versenyképességi hányados $\lceil n/(k-1)\rceil/2$ és nem is lehet jobb algoritmust adni hipergráfoknak ezen osztályán. Továbbá ez a tétel azt is bizonyítja ($k=3$ választásával), hogy ellentétben az online gráfokkal hipergráfok esetében nincs szublineáris versenyképességi hányadosú online algoritmus.

Vizsgáltuk a 2-színezhető k maximális fokú hipergráfokat is. Könnyű látni, hogy minden k maximális fokú hipergráf $k+1$-színezhető, mivel $\mathcal{FF}$ legfeljebb $k+1$ színt használva színezi őket.

27. tétel *Minden $\mathcal{A}$ online hipergráfszínező algoritmushoz és $d > 2$ egészhez létezik $\frac{(d-1)^k-1}{d-2}$ csúcsú 2-színezhető k maximális fokú H hipergráf, amelyre $\chi_{\mathcal{A}}(H) \geq k+1$.*

Tekintve hipergráfok azon osztályát, amelyben a független élrendszerek maximális elemszáma korlátos, $\mathcal{FF}$ a következő hatékonyságot tudja elérni.

28. tétel *Minden H hipergráfra $\mathcal{FF}$ a H-t legfeljebb $2 \cdot \nu(H) + 1$ színnel színezi.*

Mivel egy véges projektív sík bármely két egyenese metsző (a független élrendszerek maximális elemszáma 1), azt is kaptuk, hogy $\mathcal{FF}$ legfeljebb 3 színnel színezi a véges projektív síkokat. Másrészt amint a következő állítás mutatja, nincs olyan online algoritmus, amely kevesebb színt használna, mint $\mathcal{FF}$ ebben az esetben.

30. tétel *Nincs olyan online algoritmus, amely 3-nál kevesebb színnel színezne valamely véges projektív síkot.*

4. A k-szerver probléma

A k-szerver probléma a következő. Adott egy metrikus tér k mozgatható szerverrel, amelyek a tér különböző pontjain foglalnak helyet és a kérések (pontok) egy sorozata. Minden egyes kérést ki kell szolgálni egy szerver odamozgatásával a kért pontra. A cél minimalizálni a teljes költséget, ami a szerverek által bejárt távolságok összege. Az optimális költséget a ϱ sorozaton $\mathrm{opt}(k, \varrho)$-val jelöljük. Egy k-szerver algoritmus online, ha minden kérést azonnal kiszolgál, amikor az megérkezik (bármilyen, a jövőbeli kérésekre vonatkozó előzetes információ nélkül).

4.1. Egy véletlen algoritmus felbontható terekre

Az alfejezet eredményeit [12] tartalmazza. Ezek az eredmények [4] és [5] nyomán születtek.

A probléma egy megszorítását vizsgáltuk, nevezetesen a „μ-HST”-nek nevezett tereken kerestünk hatékony algoritmust. Ezeket a tereket [2]-ben definiálták a következőképp:

32. definíció *$\mu \geq 1$-re egy μ-hierarchikusan jólszeparálható fa (μ-HST) egy gyökeres T fa levelein definiált metrikus tér. Minden $u \in T$ csúcshoz hozzá van rendelve egy $\Lambda(u) \geq 0$ címke, amely pontosan akkor 0, ha u a T fa levele. a címkékre teljesül, hogy ha az u csúcs a v csúcs gyereke, akkor $\Lambda(u) \leq \Lambda(v)/\mu$. A távolság $x, y \in T$ csúcsok között $\Lambda(\mathrm{lca}(x, y))$-nak definiáljuk, ahol $\mathrm{lca}(x, y)$ az x és y csúcsok legkisebb közös őse a T fában.*

A μ-felbontható terek fogalma [17]-ben kerültek bevezetésre. Ennek tekintettük egy speciális esetét.

33. definíció *Legyen $\mathcal{M}$ egy metrikus tér. $\mathcal{M}$ -et* uniform módon μ-felbonthatónak *nevezzük valamely $\mu > 1$-re, ha a pontjai $t \geq 2$ blokkba partícionálhatók úgy, hogy a következő feltételek teljesüljenek:*

1. *ha $x, y \in \mathcal{M}$ különböző blokkokban vannak, akkor a távolságuk pontosan Δ, amely $\mathcal{M}$ átmérője;*
2. *minden egyes B_i blokk átmérője legfeljebb Δ/μ.*

Egy adott ϱ kéréssorozat esetén a sorozat i-edik tagját jelöli ϱ_i, és a ϱ sorozat i hosszú kezdőszeletét pedig $\varrho_{\leq i}$.

Azon pontok halmazát, amelyeken a szerverek elhelyezkednek, *konfiguráció*nak nevezzük. Adott B_s blokkra, ϱ kéréssorozatra és C kezdő konfigurációra B_s-ben jelölje $\mathcal{A}_s(C, \varrho)$ az $\mathcal{A}$ algoritmus által számított költséget ϱ azon részsorozatára, amely a B_s blokkba érkező kérésekből áll. Minden ℓ szerverszám esetén jelölje $\mathcal{A}_s(\ell, \varrho)$ a $\max_{|C|=\ell} \mathcal{A}_s(C, \varrho)$-t, ahol C végigfut az összes ℓ szerverből álló kezdőkonfiguráción $\mathcal{B}_s$-ben. Továbbá jelölje $\mathrm{opt}_s(C, \varrho)$ az optimális költséget a C kezdőkonfigurációból indulva ϱ azon részsorozatára, amely a B_s blokkba érkező kérésekből áll, és legyen $\mathrm{opt}_s(\ell, \varrho) = \min_{|C|=\ell} \mathrm{opt}_s(C, \varrho)$. Ennélfogva – ha ϱ nemüres – $\mathrm{opt}_s(0, \varrho)$-t végtelennek definiáljuk.

Algoritmusunk a következő fogalmon alapszik.

34. definíció *A B_s blokk* igénye *a ϱ kéréssorozatra*

$$D_s(\varrho) := \min\{\ell \,|\, \mathrm{opt}_s(\ell, \varrho) + \ell\Delta = \min_j\{\mathrm{opt}_s(j, \varrho) + j\Delta\}\},$$

ha ϱ nemüres, különben 0.

Bevezettünk egy technikai jelölést is.

36. definíció *Tegyük fel, hogy $\mathcal{N}$ metrikus tér, $\mathcal{A}$ egy véletlen online algoritmus, f valós függvény és $\mu > 0$ valós számok kielégítik a következő feltételeket:*

1. *$f(\ell)/\log \ell$ monoton nemcsökkenő;*
2. *minden $0 < \ell \leq \mu$ és $\mathcal{N}$-beli ϱ kéréssorozat esetén,*

$$\mathrm{E}[\mathcal{A}(\ell, \varrho)] \;\leq\; f(\ell) \cdot \mathrm{opt}(\ell, \varrho) + \frac{f(\ell) \cdot \ell \cdot \mathrm{diam}(\mathcal{N})}{\log \ell}.$$

Ekkor $\mathcal{A}$-t az $\mathcal{N}$ téren (f, μ)-hatékony algoritmusnak nevezzük.

A következő tételt bizonyítottuk:

37. tétel *Tegyük fel, hogy $\mathcal{M}$ uniform módon μ-felbontható tér és $\mathcal{A}$ egy, a tér minden blokkján (f, μ)-hatékony algoritmus. Ekkor létezik a $\mathcal{M}$ téren egy (f', μ)-hatékony algoritmus, ahol $f'(x) = c \cdot f(x) \log x$ valamely $c > 0$ abszolút konstansra.*

A tételbeli algoritmus az $\mathcal{A}$ algoritmust szubrutinként használja és *fázis*okban működik. Jelölje $\varrho^{(p)}$ a p-edik fázis kéréssorozatát. A fázisban a Shell algoritmus a következőképp működik:

$\mathcal{SH}$ algoritmus

Kezdetben megjelöljük azokat a blokkokat, amelyek nem tartalmaznak szervert.

Amikor $\varrho_i^{(p)}$, a fázis i-edik kérése a B_s blokkba érkezik, kiszámítjuk a $D_s(\varrho_{\leq i}^{(p)})$ igényt és

$$D_s^*(\varrho_i^{(p)}) = \max\{D_s(\varrho_{\leq j}^{(p)}) | j \leq i\}$$

a maximális igényt a blokkra (megjegyezzük, hogy ezek az értékek más blokkokban nem változnak).

(i) Ha $D_s^*(\varrho_i^{(p)})$ kisebb, mint a B_s-beli szerverek pillanatnyi száma, akkor a kérést $\mathcal{A}$ algoritmus szerint szolgáljuk ki a B_s blokkon belül.

(ii) Ha $D_s^*(\varrho_i^{(p)})$ egyenlővé válik B_s-beli szerverek pillanatnyi számával, akkor a kérést $\mathcal{A}$ algoritmus szerint szolgáljuk ki a B_s blokkon belül és a B_s blokkot megjelöljük.

(iii) Ha $D_s^*(\varrho_i^{(p)})$ nagyobb, mint a B_s-beli szerverek pillanatnyi száma, a B_s blokkot megjelöljük és addig hajtjuk végre ismételten az alábbi részt, amíg $D_s^*(\varrho_i^{(p)})$ szerver nem lesz a blokkban vagy nem tudjuk végrehajtani a lépéseket (ez akkor történik, ha minden blokk jelöltté vált)

> Válasszunk egy jelöletlen $B_{s'}$ blokkot véletlenül egyenletes eloszlás szerint és egy szerver ebben a blokkban szintén véletlenül. Mozgassuk a választott szervert a B_s blokkba (ezt a mozgást *ugrás*nak hívjuk) vagy a kért pontra, vagy ha ott már van szerver, akkor véletlenül válasszunk egy nem foglalt pontot B_s-ben egyenletes eloszlás szerint. Ha a $B_{s'}$-beli szerverek száma $D_{s'}^*(\varrho_i^{(p)})$ lesz az ugrás miatt, jelöljük meg azt a blokkot. Mind B_s-ben, mind $B_{s'}$-ben indítsuk újra az $\mathcal{A}$ algoritmust a blokkok aktuális konfigurációjából.

Ha nem tudjuk $D_s^*(\varrho_i^{(p)})$-ig növelni a szerverek számát a B_s blokkban a fenti lépések ismétlésével (minden blokk jelöltté vált), akkor $p+1$-edik fázist kezdjük és a legutolsó kérés már ehhez az új fázishoz tartozik.

Valamely $c \log k$-kompetitív algoritmusból indulva a 37. tétel ismételt alkalmazásával a következő eredményt kapjuk:

43. következmény *Minden h magas μ-HST-re ($\mu \geq k$) létezik $(c_1 \log k)^h$-kompetitív véletlen online algoritmus, ahol c_1 egy konstans. Következésképpen ha $h < \frac{\log k}{\log c_1 + \log\log k}$, ez az algoritmus $o(k)$-kompetitív.*

4.2. Az elutasításos k-szerver problémával kapcsolatos eredmények

Egy általánosabb modellt vizsgálunk, amelyben a kéréseket el lehet utasítani. Az alfejezet eredményei [16]-ben találhatóak. Ebben a problémában az i-edik kérés egy (ϱ_i, w_i) pár minden i-re, ahol ϱ_i egy pont $w_i > 0$ a *büntetés* az elutasítás esetén. Minden kérés kiszolgálható ugyanazon a módon, mint korábban, vagy elutasítható a kéréshez adott büntetés kifizetése mellett. Egy algoritmus költsége a szerverek által bejárt távolságok összege plusz az elutasított kérések büntetéseinek összege. A következőket bizonyítottuk.

44. tétel *Semmilyen $c < 2k$ esetén nem létezik gyengén c-kompetitív algoritmus az elutasításos k-szerver problémára uniform téren.*

Adtunk egy Threshold nevű gyengén kompetitív algoritmust az elutasításos k-szerver problémára uniform téren. A Threshold algoritmus a korábban látott jelölő eljárást alkalmazza és válogatós. Legyen $t > 0$ rögzített szám.

$\mathcal{TH}_t$ algoritmus

Minden ponthoz tartozik egy számláló. Kezdetben minden számláló értéke 0 és a pontok jelöletlenek. Minden kérés után a jelöléseket frissítjük, ezután ha szükséges, szervert mozgatunk a következőképp:

(i) Növeljük meg a kért pont számlálóját a kéréshez tartozó büntetéssel.

(ii) Ha a kért pont számlálója legalább t, jelöljük meg a pontot. Abban a pillanatban, amint $k+1$ pont jelöltté vált, minden jelölést (beleértve a legutolsót) töröljük és minden számlálót 0-ra állítunk.

(iii) Ha a kért ponton van szerver, akkor nincs szervermozgás.

(iv) Ha a kért pont jelöletlen és nincs rajta szerver, elutasítjuk.

(v) Ha a kért pont jelölt és nincs rajta szerver, akkor a legrégebben mozgott szervert mozgatjuk a kért pontra.

---

45. tétel *$\mathcal{TH}_1$ algoritmus gyengén $(2k+1)$-kompetitív uniform tereken.*

Végül a $\mathcal{TH}_t$ algoritmus véletlen változatát vizsgáltuk. Legyen $t > 0$ rögzített szám.

$\mathcal{RTH}_t$ algoritmus

Minden ponthoz tartozik egy számláló. Kezdetben minden számláló értéke 0 és a pontok jelöletlenek. Minden kérés után a jelöléseket frissítjük, ezután ha szükséges, szervert mozgatunk a következőképp:

(i) Növeljük meg a kért pont számlálóját a kéréshez tartozó büntetéssel.

(ii) Ha a kért pont számlálója legalább t, jelöljük meg a pontot. Abban a pillanatban, amint $k+1$ pont jelöltté vált, minden jelölést (beleértve a legutolsót) töröljük és minden számlálót 0-ra állítunk.

(iii) Ha a kért ponton van szerver, akkor nincs szervermozgás.

(iv) Ha a kért pont jelöletlen és nincs rajta szerver, elutasítjuk.

(v) Ha a kért pont jelölt és nincs rajta szerver, akkor véletlenül egyenletes eloszlás szerint választunk a jelöletlen pontokon lévő szerverek közül és a kért pontra mozgatjuk.

---

46. tétel *$\mathcal{RTH}_2$ algoritmus $(6\mathrm{H_k}+2)$-kompetitív uniform tereken.*

Hivatkozások

[1] N. Alon, U. Arad, Y. Azar, Independent Sets in Hypergraphs with Applications to Routing Via Fixed Paths, *Proc. of APPROX 99, LNCS 1671* (1999), 16–27.

[2] Y. Bartal, B. Bollobás, M. Mendel, A Ramsey-type theorem for metric spaces and its application for metrical task system and related problems, *Journal of Computer and System Sciences* **72** (2006), 890–921.

[3] Y. Bartal, S. Leonardi, A. Marchetti-Spaccamela, J. Sgall, L. Stougie, Multiprocessor scheduling with rejection, *SIAM Journal on Discrete Mathematics* **13** (2000) 64–78.

[4] Y. Bartal, M. Mendel, Randomized k-server algorithms for growth-rate bounded graphs, *Journal of Algorithms* **55** (2005), 192–202.

[5] B. Csaba, S. Lodha, A randomized on-line algorithm for the k-server problem on a line, *Random Structures and Algorithms* **29** (2006), 82–104.

[6] B. Csaba, A. Pluhár, A randomized algorithm for on-line weighted bipartite matching problem, *Journal of Scheduling* **11** (2008), 449–455.

[7] R.L. Graham, Bounds for certain multiprocessor anomalies, *Bell System Technical Journal* **45** (1966), 1563–1581.

[8] Cs. Imreh, J. Noga, Scheduling with Machine Cost, *Proc. APPROX'99, Lecture Notes in Computer Science*, Vol. 1761, Springer, Berlin, 1999, 168–176.

[9] H. A. Kierstead, Coloring Graphs On-line, in: A. Fiat, and G. J. Woeginger (eds.) *Online algorithms: The State of the Art, LNCS 1442*, 1998, 281–305.

[10] H. A. Kierstead, On-line coloring k-colorable graphs, *Israel Journal of Mathematics* **105** (1998), 93–104.

[11] L. Lovász, M. Saks, W. T. Trotter, An on-line graph coloring algorithm with sublinear performance ratio, *Discrete Mathematics* **75** (1989), 319–325.

[12] J. Nagy-György, Randomized algorithm for the k-server problem on decomposable spaces, *Journal of Discrete Algorithms* (2009), doi: 10.1016/j.jda.2009.02.005

[13] J. Nagy-György, Online coloring graphs with high girth and high oddgirth, submitted

[14] J. Nagy-György, Cs. Imreh, Online scheduling with machine cost and rejection, *Discrete Applied Mathematics* **155** (2007), 2546–2554.

[15] J. Nagy-György, Cs. Imreh, Online hypergraph coloring, *Information Processing Letters* **109** (2008), 23–26.

[16] J. Nagy-György, Cs. Imreh, k-server problem with rejection, manuscript

[17] S. S. Seiden, A general decomposition theorem for the k-server problem, *Information and Computation* **174** (2002), 193–202.

[18] J. Sgall, On-line scheduling, in: A. Fiat, G.J. Woeginger (eds.) *Online algorithms: The State of the Art, Lecture Notes of Computer Science*, Vol. 1442, Springer-Verlag Berlin, 1998, 196–231.