Véletlenített online algoritmusok

A gyakorlati problémákban gyakran fordulnak elő olyan optimalizálási feladatok, ahol a bemenetet csak részenként ismerjük meg, és a döntéseinket a már megkapott információk alapján, a további adatok ismerete nélkül kell meghoznunk.

Ilyen feladatok esetén *on-line problémáról* beszélünk. Az on-line algoritmusok elméletének igen sok alkalmazása van a számítástudomány, a közgazdaságtan, és az operációkutatás különböző területein. Mivel egy on-line algoritmusnak részenként kell meghozni a döntéseit a teljes bemenet ismerete nélkül, ezért egy ilyen algoritmustól nem várhatjuk el, hogy a teljes információval rendelkező algoritmusok által megkapható optimális megoldást szolgáltassa. Azon algoritmusokat, amelyek ismerik a teljes inputot off-line algoritmusoknak nevezzük. A legelterjedtebb módszer online algoritmusok elemzésére a versenyképességi elemzés.

A továbbiakban egy tetszőleges ALG on-line algoritmusra az I inputon felvett célfüggvényértéket $ALG(I)$-vel jelöljük. Az I inputon felvett optimális off-line célfüggvényértéket $OPT(I)$-vel jelöljük. Használva ezt a jelölésrendszert a versenyképességet minimalizálási problémákra a következőképpen definiálhatjuk. Egy minimalizálási probléma esetén az ALG algoritmus C-versenyképes ha van olyan B konstans, hogy $ALG(I) \leq C \cdot OPT(I) + B$ teljesül minden I input esetén. Egy algoritmus versenyképességi hányadosa a legkisebb olyan C szám, amelyre az algoritmus C-versenyképes. Szokás használni a definiciót a B additív konstans nélkül is, ez esetben abszolút versenyképességi hányadosról beszélünk. Megjegyezzük, hogy a fentiekben definiált versenyképességi hányadost szokták enyhe versenyképességi hányadosnak is nevezni.

Véletlenített algoritmus esetén $ALG(i)$ nem egy konstans, hanem egy valószíq-nűségi változó, amely az algoritmus véletlen döntéseitől függ. Véletlenített algoritmusoknál a versenyképesség definíciójában ALG(I) várható értékét használjuk. Tehát egy véletlenített ALG algoritmus C-versenyképes ha van olyan B konstans, hogy $E(ALG(I)) \leq C \cdot OPT(I) + B$ teljesül minden I input esetén. Megkülönböztetünk 3 különböző változatot attól függően milyen információkkal rendelkezik az inputot generáló ellenfél. Hanyag ellenfélről beszélünk abban az esetben ha az ellenfélnek nincs információja az algoritmus véletlen döntésinek kimeneteiről, más szóval az ellenfél előre az algoritmus futása előtt véglegesen definiálja az inputot. Adaptív ellenfélről beszélünk abban az esetben ha az ellenfél az input következő részét mindig az algoritmus aktuális döntései után definiálja. Ebben az esetben megkülönböztetünk adaptív online és adaptív offline ellenfeleket. Az adaptív online ellenfél a problémát online oldja meg, mindig a már definiált adatok alapján dönt, az adaptív offline ellenfél pedig az eljárás végén a teljes input ismeretében oldja meg a feladatot.

Lapozási probléma

A lapozási problémában a számítógépek gyorsmemóriájának a kezelését modellezi. Adott lapok egy univerzuma, amely lapok egy sorozata az input. Az algoritmusnak egy k lap kapacítású gyorsmemóriát kell kezelnie. Ha az aktuálisan igényelt lap nincs a memóriában, akkor a lapot az algoritmusnak be kell tennie és ehhez, amennyiben már nincs hely, valamely lapot ki kell raknia. Ezt

az esenényt, amikor egy igényelt lap nincs a memóriában hibának hívjuk, és a cél a hibák számának minimalizálása. A probléma online, ami azt jelenti, hogy az algoritmusnak a lap elhelyezéséhez szükséges döntést (valamely lap kivétele a memóriából) a további igények ismerete nélkül kell meghozni. A következő állítás azt mutatja, hogy nincs kicsi versenyképességi hányadossal rendelkező determinisztikus algoritmus a feladat megoldására.

Tétel Nincs olyan determinisztikus algoritmus a lapozási problémára, amely versenyképességi hányadosa kisebb, mint k.

Bizonyítás Vegyünk k+1 lapot és legyen A egy tetszőleges online algoritmus. Legyen L_n az az n elemből álló input, amelyben az új elem mindig az a lap, amely lap éppen hiányzik A memóriájából. Legyen továbbá n osztható k-val. Ekkor A minden lapnál hibázik, így $A(L_n) = n$. Legyen LFD (longest forward distance) az az offline algoritmus, amely ha egy lapot ki kell rakni a memóriából mindig azt a lapot választja a lapok közül, amelyre a következő igény a legkésőbb érkezik. Fontosnak tarjuk megjegyezni, hogy valójában LFD az optimális offline megoldást adja, de ezt nem igazoljuk, mivel a bizonyítás során az algoritmus optimalitására nincs szükség. Vegyük észre, hogy amennyiben LFD hibázik, akkor a következő k-1 kérés során nem hibázhat újra. Ez a tulajdonság azért teljesül, mert az algoritmus következő hibája, akkor következik be, amikor azt a lapot igényeljük, amelyet kitett a memóriából. És az LFD szabály miatt ezt a kérést meg kell, hogy előzze a másik k-1 lapra vonatkozó kérés. Tehát azt kaptuk, hogy $LFD(L_n) \leq n/k$. Másrészt $OPT(L_n) \leq LFD(L_n)$, tehát $A(L_n)/OPT(L_n) \geq n/(n/k)$, amivel a tétel állítását igazoltuk (az additív kostans nem tehet lehetővé k-nál kisebb versenyképességi hányadost, mert tetszőlegesen hosszú sorozatot vehetünk).

Másrészt számos k-versenyképes determinisztikus algoritmus van, itt k-versenyképes algoritmusok egy osztályát a bélyegző algoritmusokat ismertetjük, mivel ezek lesznek azok, amelyeket véletlenített algoritmussá fejlesztünk tovább.

Bélyegző (továbbiakban B) algoritmus

Az algoritmus bélyegeket használ a memóriában. Kezdetben egyetlen lap sincs megjelölve. Egy kérés érkezésekor az algoritmus a következő lépéseket hajtja végre.

- 1. Ha a kért lap a memóriában van, akkor amennyiben még jelöletlen megjelöljük.
- 2. Ha a kért lap nincs a memóriában, és nincs már jelöletlen lap a memóriában, akkor az összes jelölést töröljük. Ezt követően veszünk egy jelöletlen lapot a memóriából (az előző lépés miatt van ilyen) a kért lapot ennek a lapnak a helyére berakjuk, majd megjelöljük.

Az algoritmus viselkedése nagymértékben függ attól, hogy milyen szabály alapján választjuk ki a törlendő lapot, de a következő tétel mutatja, hogy a versenyképességi hányadosa nem.

Tétel A B algoritmus versenyképességi hányadosa k.

Bizonyítás A tétel bizonyításához elegendő belátni, hogy az algoritmus k-versenyképes, az általános alsó korlát alapján adódik, hogy nem lehet jobb. Vegyünk egy tetszőleges inputot, jelölje I. Bontsuk fel ezt az inputot fázisokra a következőképpen. Az első fázis kezdődjön, az első elemnél. Ezt követően minden egyes fázis a megelőző fázis utolsó eleme után jövő elemnél kezdődik, és a leghosszabb olyan sorozatát tartalmazza a kéréseknek, amelyek legfeljebb k különböző lapot igényelnek. (Tehát a következő fázis akkor kezdődik, amikor a k+1-edik különböző lap megjelenik.) Érdemes megjegyezni, hogy a fázis a bélyegek kiosztásával is definiálható, akkor kezdődik új fázis, amikor a B algoritmus törli a bélyegeket a memóriában. Az algoritmus definíciójából következik, hogy B legfeljebb k hibát vét egy fázis alatt, (ha egy lapon hibázik, azon többet már nem fog a fázisban, hiszen megjelölte). Most vizsgáljuk az optimális offline algoritmust. Egy tetszőleges fázis második kérésénél az offline memóriában benne van a fázis első eleme. Ezt követően a következő fázis első eleméig k további elem jelenik meg, így az offline algoritmusnak legalább egy hibát kell vétenie, az adott fázis második elemétől, a következő fázis első eleméig tartó kéréssorozaton. Következésképpen minden fázishoz (kivéve esetleg az utolsót) hozzárendeltük az offline algoritmus egy egy hibáját. Jelölje r a fázisok számát. Ekkor $B(I) \leq rk+k-1$ és $OPT(I) \geq r$, következésképp $B(I) \leq kOPT(I)+k-1$, amivel a tétel állítását igazoltuk.

A bélyegző algoritmus véletlenített változatával, amelyben nem determinisztikus döntés alapján választjuk a lapot, amely kikerül a memóriából jobb eredményt érhetünk el. A véletlenített bélyegző (RB) algoritmust a következőképpen definiálhatjuk.

Az RB algoritmus

- 1. Ha a kért lap a memóriában van, akkor amennyiben még jelöletlen megjelöljük.
- 2. Ha a kért lap nincs a memóriában, és nincs már jelöletlen lap a memóriában, akkor az összes jelölést töröljük. Ezt követően veszünk egy jelöletlen lapot egyenletes elszolás alapján a memóriából (az előző lépés miatt van ilyen) a kért lapot ennek a lapnak a helyére berakjuk, majd megjelöljük.

Tétel Az RB algoritmus $2H_k$-versenyképes hanyag ellenfél ellen.

Bizonyítás Vegyünk egy tetszőleges I inputot, és rögzítsünk egy optimális offline algoritmus, amit jelöljön OFF. Az inputot bontsuk fázisokra ugyanúgy, mint a determinisztikus esetben. Ekkor ismét teljesül, hogy akkor kezdődik új fázis, amikor az RB algoritmus törli a bélyegeket a memóriában. Vegyük észre, hogy a fázisok nem függnek az algoritmus véletlen döntéseitől (a fázison belüli költsége igen). Jelölje a fázisok számát n. Most vizsgáljuk meg, hogy egy adott i-re mennyi RB várható költsége az i-edik fázisban. Ennek meghatározásához a fázis során kért lapokat két halmazba osztjuk. Réginek nevezzük azokat a lapok, akik a fázis megkezdésekor RB memóriájában voltak, ezek számát jelölje r_i A többi lapot (akik nem szerepeltek RB memóriájában) új lapnak nevezzük, ezek száma legyen u_i. Minden új lap hibát okoz, és egyetlen lap sem okoz egynél több hibát egy fázison belül, így azt kapjuk, hogy az új lapok által kapott hibák

száma u_i. Most vizsgáljuk meg, hogy mennyi a régi lapok által okozott várható hibák száma. Nyilvánvalón egy régi lap csak az első megjelenésénél okozhat hibát, pontosan, akkor ha a lapot az algoritmus már kirakta a memóriájából. Vizsgáljuk meg ennek a valószínűségét. Tegyük fel, hogy egy régi lap első megjelenésénél, a memória tartalmaz x darab új lapot és y darab megjelölt régi lapot (ezeket már szerepeltek a fázisban). Ekkor a k-y jelöletlen régi lap közül egyenletes eloszlás alapján x darab nem szerepel a memóriában, így annak a valószínűsége, hogy a kért lap nem szerepel, azaz hibát okoz $x/(k-y)$. Tehát a lap általi hibák számának várható értéke $x/(k-y) \leq u_i/(k-y)$. Következésképpen a régi lapok által várható hibák száma legfeljebb $\sum_{j=0}^{r_i} u_i/(k-j)$. Így ízt kapjuk, hogy az algoritmus várható költsége a fázis során legfeljebb

$$u_i + \sum_{j=0}^{r_i} u_i/(k-j) \leq u_i H_k.$$

Összefoglalva azt kaptuk, hogy $RB(I) \leq \sum_{i=1}^{n} u_i H_k$.

Most vizsgáljuk meg az optimális költséget. Ehhez legyen d_i azon lapok száma, amelyek az i-edik fázis előtt szerepelnek az offline memóriában, de nem szerepelnek RB memóriájában. Feltesszük, hogy $d_1 = 0$, azaz ugyanazzal a memóriával kezdenek az algoritmusok. Használjuk a d_{n+1} értéket is, ez az algoritmus befejezésekor fennálló érték. Legyen az offline algoritmus költsége az i-edik fázisban OFF_i. Mivel az i-edik fázisban pontosan azokat a lapokat kérik, amelyek szerepelnek a végén RB memóriájában, ezért minden lap, ami a fázis végén ezektől különbözik egy hibát okoz OFF számára. Következésképpen $OFF_i \geq d_{i+1}$. Másrészt azon új lapok, amelyek nem voltak OFF memóriájában a fázis előtt, OFF számára is hibát okoznak. Ezen lapok száma legalább $u_i - d_i$, így azt kapjuk, hogy $OFF_i \geq u_i - d_i$. A két korlát alapján adódik, hogy $OFF_i \geq (u_i - d_i + d_{i+1})/2$. Összegezve ezeket az értékeket adódik, hogy $OFF(I) \geq (\sum_{i=1}^{n} u_i - d_1 + d_{n+1})/2 \geq (\sum_{i=1}^{n} u_i)/2$.

Következésképpen azt kapjuk, hogy $RB(I) \leq 2H_k OFF(I)$, amivel a tételt igazoltuk.

A következő tétel azt mutatja, hogy nem adható aszimptotikusan jobb algoritmus hanyag ellenfél ellen, mint RB.

Tétel Nincs olyan véletlenített algoritmus a lapozási problémára, amely veresnyképességi hányadosa hanyag ellenfél ellen kisebb min H_k.

Bizonyítás A tétel bizonyításához a Yao elvet használjuk fel. Tehát vehetünk egy tetszőleges valószínűségi eloszlást, és az azáltal generált inputon kell megvizsgálnunk a legjobb determinisztikus online algoritmust. Általában egy adott eloszlásra a legjobb determinisztikus algoritmus megtalálása igen nehéz feladat, így az ilyen esetekben gyakran használt ötlet olyan eloszlást választani, amelyen minden algoritmus ugyanazt a várható erdményt éri el. Most is ezt tesszük. Vegyünk k+1 különböző lapot és legyen I kérések egy olyan sorozata, amelyben n darab kérés érkezik, melyek mindegyike egyenletes eloszlás alapján választ egy lapot a $k+1$ lap közül. Definiáljuk a generált sorozat fázisait, az előző bizonyításokhoz hasonlóan. Az i-edik fázis az LFD algoritmus i-edik hibájánál kezdődik és a következő hibát megelőző lapig tart. Ekkor LFD költsége minden

fázisban 1. Most vizsgáljuk az adott fázis várható hosszát. Akkor következik be a következő hiba, amikor arra a lapra érkezik a kérés, amit LFD kitett a memóriából, de ezt megelőzően a többi lapra is kellett kéréseknek érkeznie. Tehát egy fázis várható hossza 1-el rövidebb a legrövidebb olyan sorozat várható hosszánál, amelyben mind a k+1 lapra érkezik kérés. Másrészt pontosan az ilyen sorozatok várható hosszát vizsgáltuk a kupon gyűjtő problémában (a ládák, amibe a kupont rakjuk a lapok, amelyeket egyenletes eloszlás alapján generáltunk). Következésképpen egy ilyen legrövidebb sorozat várható hossza $(k+1)H_{k+1}$. Azaz egy fázis várható hossza $(k+1)H_{k+1} - 1 = (k+1)H_k$.

Most vegyük észre, hogy tetszőleges online algoritmust véve minden lépésben a hiba valószínűsége, így várható értéke $1/(k+1)$. Tehát az online algoritmus várható költsége egy fázisban $(k+1)H_k/(k+1) = H_k$. Következésképpen minden fázisra az online algoritmus várható költsége H_k-szor az offline költség, amivel az állítást igazoltuk.