Szegedi Tudományegyetem
Számítógépes Algoritmusok és Mesterséges Intelligencia Tanszék

Online előrenéző és paraméter tanuló algoritmusok nyugtázási és ütemezési problémákra

Ph.D. értekezés tézisei

Németh Tamás

Témavezető:
Dr. Imreh Csanád
tanszékvezető egyetemi docens

SZTE TTIK, Informatika Doktori Iskola

Szeged
2013

1. Bevezetés

On-line problémáról beszélünk az olyan optimalizálási feladatok esetében, ahol a bemenetet csak részenként ismerjük meg, és a döntéseinket a már megkapott információ alapján, a további adatok ismerete nélkül kell meghoznunk. Egy ilyen algoritmustól nem várhatjuk el, hogy a teljes információval rendelkező algoritmusok által megkapható optimális megoldást szolgáltassa. Azon algoritmusokat, amelyek ismerik a teljes bemenetet, off-line algoritmusoknak nevezzük. Az on-line algoritmusok hatékonyságának vizsgálatára két alapvető módszert használnak. Az egyik lehetőség az átlagos eset elemzése. Ebben az esetben fel kell tételeznünk valamilyen valószínűségi eloszlást a lehetséges bemenetek terén, és a célfüggvénynek az erre az eloszlásra vonatkozó várható értékét vizsgáljuk. A másik megközelítés egy legrosszabb eset elemzés, amelyet versenyképességi elemzésnek nevezünk. Ebben az esetben az on-line algoritmus által kapott megoldás célfüggvényértékét hasonlítjuk össze az optimális off-line célfüggvényértékkel.

A továbbiakban egy tetszőleges ALG on-line algoritmusra az I bemeneten felvett célfüggvényértéket $ALG(I)$-vel jelöljük. Az I bemeneten felvett optimális off-line célfüggvényértéket $OPT(I)$-vel jelöljük. Ezt a jelölésrendszert használva a versenyképességet minimalizálási problémákra a következőképpen definiálhatjuk. Az ALG algoritmus C-versenyképes, ha $ALG(I) \leq C \cdot OPT(I)$ teljesül minden I bemenet esetén. Egy algoritmus enyhe versenyképességi hányadosa a legkisebb olyan C szám, amelyre az algoritmus enyhén C-versenyképes.

2. A nyugtázási probléma

Ebben a fejezetben összefoglaljuk a nyugtázási probléma előrenéző változatának megoldásához készített algoritmusokat, és azok versenyképességi elemzését, mely eredményeket a [7] cikkben publikáltuk.

Az informatikai hálózatokon a kommunikáció során elküldött információ csomagok formájában továbbítódik. Az esetek többségében azonban a kommunikációs csatorna nem teljesen megbízható, így az egyes információcsomagok megérkezéséről nyugtát kell küldeni. A TCP implementációk is küldenek nyugtát a csomagok fogadásakor [12]. A nyugtázási probléma tárgyalása során azt próbáljuk meghatározni, hogy melyik időpontban érdemes egy nyugtát küldeni. Abból indulunk ki, hogy egy nyugta több csomag

megérkezését tudja igazolni, azonban ha túl sokáig várunk egy-egy nyugta elküldésével az a csomag újraküldéséhez vezethet, ami többletterhelést ró a hálózatra, míg ha minden csomagot azonnal nyugtázunk akkor maguk a nyugták terhelik feleslegesen a hálózatot.

A nyugtázási probléma matematikai modelljében a bemenetet a csomagok $a_1, ..., a_n$ érkezési idejei adják. Az algoritmusnak meg kell határoznia, hogy mikor küld nyugtákat, ezeket az időpontokat $t_1, ..., t_k$ jelöli. A nyugtázási probléma költségfüggvénye $\gamma k + (1-\gamma)\sum_{j=1}^{k} L_j$, ahol k a nyugták száma és az f_{max} modellben $L_j = \max_{t_{j-1}<a_i\leq t_j}(t_j - a_i)$, a j-edik nyugta által összegyűjtött maximális késedelem, míg az f_{sum} modellben $L_j = \sum_{t_{j-1}<a_i\leq t_j}(t_j - a_i)$ a j-edik nyugta által összegyűjtött teljes késedelem.

Ha az algoritmus ismeri az érkezési időket a következő c hosszúságú időintervallumra azt idő előretekintő tulajdonságnak nevezzük. Ha $c < \gamma/(1-\gamma)$ az ébresztő algoritmus egy kiterjesztettebb verzióját használjuk, a továbbiakban TLA algoritmusnak nevezzük (Time Lookhead Alarming Algorithm). Az f_{max} célfüggvény esetén az a_j időpontban beállítjuk az ébresztőt $a_j + \gamma/(1-\gamma) - c$ időpontra. Ha az a_{j+1} csomag az ébresztési időpont előtt megérkezik vagy ha az a_{j+1} csomagot az $a_j + \gamma/(1-\gamma) - c$ időpontban az $(a_j + \gamma/(1-\gamma) - c, a_j + \gamma/(1-\gamma)]$ intervallumban látjuk (előre) akkor átállítjuk az ébresztőt az $a_{j+1} + \gamma/(1-\gamma) - c$ időpontra. Ellenkező esetben, azaz nem érkezik a következő csomag a $(a_j, a_j + \gamma/(1-\gamma)]$ időintervallumban, nyugtát küldünk az $a_j + \gamma/(1-\gamma) - c$ időpontban, mely az összes még nyugtázatlan csomagot nyugtázza. Az f_{sum} célfüggvény esetén pedig az a_j időpontban beállítjuk az ébresztőt az $a_j + e_j$ időpontra, ahol $e_j = (\gamma/(1-\gamma) - \sum_{a_i\in\sigma_j}(a_j - a_i))/|\sigma_j|$. Ha a következő csomag megérkezik az $\max\{a_j, a_j + e_j - c\}$ időpont előtt, vagy látjuk a_{j+1}-et az előrenéző intervallumban, átállítjuk az ébresztőt. Egyébként (nincs csomag az $(a_j, a_j + e_j]$) intervallumban) nyugtát küldünk a $\max\{a_j, a_j + e_j - c\}$ időpontban, ami minden nyugtázatlan csomagot nyugtáz.

1. Tézis

1. Tétel. *Az f_{max} modellben a TLA algoritmus $max\{1, 2-\frac{1-\gamma}{\gamma}c\}$-versenyképes.*

2. Tétel. *Nem létezik olyan c-előrenéző algoritmus az $f_{\max}$ modellben, amelynek kisebb a versenyképességi hányadosa, mint* $\max\{1, 2 - \frac{1-\gamma}{\gamma}c\}$.

3. Tétel. *Az f_{sum} modellben, tetszőleges c-re a TLA algoritmus versenyképességi hányadosa* 2.

Ha $c > \gamma/(1-\gamma)$ 2-nél kisebb versenyképességű algoritmust is adhatunk. A LIP algoritmus (Lookahead Interval Planning Algorithm) a következő. Az algoritmus blokkokra bontja az inputot, majd minden blokkra meghatározza a nyugták küldésének időpontját az optimális off-line algoritmus segítségével. Egy-egy blokk mindig az első nyugtázatlan csomaggal kezdődik. Az algoritmus először meghatározza, hogy van-e 2 egymást követő csomag a c hosszúságú előretekintő intervallumban melyre $a_{i+1} - a_i > \gamma/(1-\gamma)$. Ha van ilyen pár, a blokkot lezárjuk az első ilyen a_i időpontnál, egyébként a blokk intervallumának hossza c. Ez után az algoritmus meghatározza a blokkra a nyugták küldésének időpontját az optimális megoldáshoz az off-line probléma optimális megoldásával [6], majd elküldi a nyugtákat a megoldás szerint és áttér a következő blokkra.

2. Tézis

4. Tétel. *A LIP algoritmus* $1 + \frac{\gamma}{(1-\gamma)c}$*-versenyképes.*

5. Tétel. *Nem létezik online algoritmus az* f_{sum} *modellben, amelynek kisebb a versenyképességi hányadosa, mint* $1 + \Omega(1/c^2)$*, ha* $c > \gamma/(1-\gamma)$*.*

6. Tétel. *Nem létezik olyan c-előrenéző semi-on-line algoritmus a nyugtázási problémára az* f_{sum} *modellben, amelynek kisebb a versenyképességi hányadosa, mint* $2\gamma/(c(1-\gamma)+\gamma)$*, ha* $c \leq \gamma/(1-\gamma)$

3. Paraméter tanuló algoritmusok a nyugtázási problémához

Ebben a fejezetben egy új algoritmust mutatunk be az on-line és félig on-line nyugtázási problémára az f_{sum} modellben, melynek átlagos hatékonysága jobb, mint a szakirodalomból ismert többi algoritmusé. Az eredményeket a [8] és [9] cikkekben publikáltuk.

Az algoritmus szakaszokban működik és az Alarm vagy a TLA algoritmust használja a csomagok nyugtázására. Először definiáljuk az ébresztő algoritmus egy verzióját melyet használni fogunk, az $Alarm_p$ algoritmust, mely egy pozitív p paramétert használ. $Alarm_p$ algoritmus $e_j = (p\gamma/(1-\gamma) - \sum_{a_i \in \sigma_j}(a_j - a_i))/|\sigma_j|$ értéket használja minden j-hez. Ez az e_j érték azt jelenti, hogy ha nem érkezik új csomag $a_j + e_j$ időpontig, az algoritmus $a_j + e_j$ időpontban nyugtázza az összes még nyugtázatlan csomagot. A késedelmi költség $\sum_{a_i \in \sigma_j}(a_j - a_i)) + |\sigma_j| \cdot e_j = p \cdot \gamma/(1-\gamma)$.

Meg kell jegyeznünk, hogy $Alarm_1$ algoritmus 2-versenyképes [6]. A TLA algoritmus teljesen hasonló kiterjesztését TLA_p jelöli.

Az új paraméter tanuló algoritmus két paraméter tanulásán alapszik, melyek r és q. Az algoritmus a következőképpen működik.

LearnAlarm(r,q)/LearnTLA(r,q) algoritmus

- *1. szakasz* Használjuk az $Alarm_1/TLA_1$ algoritmust az első r csomag nyugtázásához. Ugrás a 2. szakaszra.
- *j. szakasz ($j > 1$)*
  - Jelölje I_j az inputot, ami az utolsó $r \cdot q$ csomag. Ha kevesebb csomag érkezett, akkor I_j mindet tartalmazza.
  - A SimpleOpt algoritmust használjuk a p azon értékének meghatározására, mely minimalizálja az $Alarm_p/TLA_p$ algoritmus költségét az I_j inputon. Jelölje ezt az értéket p^*.
  - Az $Alarm_{p^*}/TLA_{p^*}$ algoritmust használjuk a következő r csomag nyugtázására majd folytassuk a $j+1$. szakaszon.

Észrevehetjük, hogy p optimális értéke olyan, hogy a nyugtázás időpontja pontosan valamely csomag beérkezési időpontjával egyenlő. Az alábbi algoritmus csak azokat az értékeket vizsgálja, melyek ennek a kritériumnak megfelelnek.

A SimpleOptAlarm Algoritmus

- *Inicializálás:* Legyen $p^* = 1$ és $Z = \infty$
- *Keresés:* Minden $1 \leq i < j \leq n$ -re
  - Legyen $s := \sum_{k=i}^{j}(a_j - a_i)$, és $p := (1-\gamma)s/\gamma$.
  - Futtassuk p-re $Alarm_p/TLA_p$-t az inputon és határozzuk meg a nyugtát számát (k)
  - Ha $k\gamma(1+p) < Z$, akkor $p^* := p$ és $Z := k\gamma(1+p)$
- Return p^*

Az algorimust teszteknek vetettük alá, hogy a hatékonyságát értékeljük. Az eredmények alapján a következő következtetéseket vonhatjuk le:

- A legfontosabb megfigyelésünk, hogy a tesztek eredményei tisztán mutatják, hogy a paraméter tanulás jelentős hatékonyságjavulást eredményez az *Alarm* algoritmus működésében.
- A teszteredmények alapján az is jól látható, hogy az új algoritmus igen érzékeny a paraméterek beállításaira. Egy pontos paraméter beállítás tovább csökkenti az ALG/OPT hányadost. Az is jól látszik, hogy a paraméterek optimális beállítása az input függvénye, de néhány általános következtetést levonhatunk. Úgy tűnik, hogy egy közepes hosszúságú tanulási szakasz jobb eredményt ad, mint egy rövid, vagy egy hosszú.
- Miután összehasonlítjuk a különböző tesztesetek eredményeit azt a következtetést is levonhatjuk, hogy az algoritmusok hasonló viselkedést mutatnak.

3. Tézis

Egy új, paraméter tanuláson alapuló algoritmust fejlesztettünk az online nyugtázási feladat megoldására és annak előrenéző változatához is. Véletlen eloszlás alapján generált és valós inputokon futtatott tesztek alapján megállapítottuk, hogy az új algoritmus lényegesen jobb hatékonyságú, mint a korábban ismert algoritmusok.

4. Paraméter tanuló algoritmus az elutasításos on-line ütemezéshez

Ebben a fejezetben az elutasításos on-line ütemezéshez mutatunk be egy új algoritmust, melyet a [10] cikkben publikáltunk.

Ez elutasításos ütemezés problémájában [2] az algoritmusnak lehetősége van az egyes munkákat elutasítani. A munkák jellemzője a végrehajtási idő és az elutasítás büntetése. Célunk az elfogadott munkák végrehajtási idejének és az elutasítás büntetéseinek összegét minimalizálni. Létezik egy 2.618-versenyképes algoritmus az online esetre tetszőleges számú géphez. Ezt az algoritmust nevezzük RTP algoritmusnak (Reject Total Penalty), melynek az egyik alapötlete az, hogy minden munkára hasonlítsuk össze a büntetést és a terhelést, majd utasítsuk el azokat a munkákat, melyekre a büntetés kisebb mint a terhelés. Ez az alapötlet még csak a mohó algoritmust adja, ami akkor hoz rossz döntést, ha a gépek száma nagy, mert ez

lehetővé teszi, hogy úgy tűnjön, hogy egy nagy munka kis terhelést jelent. Az RTP algoritmus ezeket a munkákat sokkal óvatosabban kezeli.

Algorithm RTP(α)

- 1. *Inicializáció.* Legyen $R := \emptyset$.
- 2. Egy új j munka érkezésekor
  - (i) Ha $w_j \leq \frac{p_j}{m}$, elutasítjuk a munkát.
  - (ii) Legyen $r = \sum_{i \in R} w_i + w_j$. Ha $r \leq \alpha \cdot p_j$, elutasítjuk a j munkát és $R := R \cup \{j\}$.
  - (iii) Egyébként elfogadjuk j-t és ütemezzük az első felszabaduló gépre.

1. Prepozíció. *Az RTP az* $\alpha = (\sqrt(5) - 1)/2$ *paraméterrel* $(3 + \sqrt(5))/2$ *versenyképes.*

2. Prepozíció. *Nem létezik olyan* β*-versenyképes online algoritmus tetszőleges* m*-re, ahol* $\beta < (3 + \sqrt{5})/2$

Paramétertanuló algoritmus

Bemutatunk egy új algoritmust, PAROLE (PARameter Online LEarning), mely a futása során megpróbálja megtanulni a legjobb paramétert. Az algoritmus fázisokban működik, minden fázis után választ egy új paramétert az input addig megismert része alapján. Először egy keretalgoritmust definiálunk, mely az új paramétert választó algoritmust eljárásként használja, majd megadjuk azt az eljárást is, mely a paraméter optimális értékét határozza meg az adott fázison. A fázisokat a beérkezett munkák száma alapján határozzuk meg, a fázis a 250. beérkezett munkával ér véget.

A PAROLE algoritmus (i. fázis)

- Az i fázis kezdetekor a CHOOSE algoritmus meghatározza az α_i paraméter értékét.
- Hajtsuk végre $RTP(\alpha_i)$ algoritmust az inputon már ismert részén, R halmazt is módosítjuk.
- Használjuk $RTP(\alpha_i)$-t a fázis alatt beérkező munkákra.

Azt az α_i értéket szeretnénk használni, mely mellett az $RTP(\alpha_i)(I)$ költség minimális az input már ismert részére. Úgy sejtjük, hogy ezen érték meghatározása NP-nehéz, ezért a következő mintavételező algoritmust alkalmazzuk a paraméter új értékének meghatározására. Az algoritmus a paraméter megelőző értékét is felhasználja, amit α^*-al jelölünk.

A CHOOSE algoritmus

- Generálunk egy-egy értéket egyenletes eloszlással az $[(i-1)/10, i/10]$, $i = 1, \ldots, 10$ intervallumokból. A kiválasztott értékek halmazát jelöljük S_1-el. Válasszuk ki az S_1 halmaz azon α elemét, melyre $RTP(\alpha)$-nak a legkisebb költsége van I-n. Jelöljük ezt az értéket $\bar{\alpha}$-val.

- Generálunk egy-egy értéket egyenletes eloszlással az $[\alpha^\star - i/100, \alpha^\star - (i-1)/100]$, $[\alpha^\star + (i-1)/100, \alpha^\star + i/100]$, $[\bar{\alpha} - i/100, \bar{\alpha} - (i-1)/100]$, $[\bar{\alpha} + (i-1)/100, \bar{\alpha} + i/100]$ intervallumokból is, ahol $i = 1, \ldots 10$. A kiválasztott értékek halmazát jelöljük S_2-vel. Legyen α az $S_2 \cup \{\alpha^*\} \cup \{\bar{\alpha}\}$ halmaz azon eleme melyre I inputra az $RTP(\alpha)$-nak a legkisebb költsége van.

A CHOOSE alapötlete, hogy válasszunk egy jó paraméterértéket. Két jelölt szomszédságát vizsgáljuk meg, az egyik az előző paraméter értéke, a másik egy további $\bar{\alpha}$ érték, mely a legjobb néhány olyan érték közül, melyeket az paraméter előző értékétől függetlenül választottunk.

4. Tézis

Egy új, paraméter tanuláson alapuló algoritmust fejlesztettünk az on-line elutasításos ütemezés megoldására. Véletlen eloszlás alapján generált és valós inputokon futtatott tesztek segítségével megállapítottuk, hogy az új algoritmus hasonló hatékonyságú mint a korábbi algoritmusok, viszont akkor is jól használható, ha nincs előzetes információnk az inputról.

5. On-line klaszterezés egy RTLS rendszerben

Ebben a fejezetben bemutatjuk az on-line klaszterezési eljárásunkat és eredményét egy valós idejű helymeghatározási rendszer alkalmazásában. A rendszer az eredmények felhasználásával az objektumok gyors, nagy pontosságú pozíció-meghatározását végzi el. Az eredményeket a [11] cikkben publikáltuk.

A valós idejű helymeghatározó rendszerek feladata a különböző objektumok pozíciójának nagy pontosságú meghatározása A rendszerben az objektumok olyan miniatűr rádióadókkal vannak felszerelve, melyek egy az Intézet által kifejlesztett beltéri működésű valós idejű pozíciómeghatározási rendszer (WISMIT) által feldolgozható periodikusan sugárzott rádiójeleket küldenek. Ezeket a rádiójeleket fogják az infrastruktúra különböző mérőállomásai, melyek a jelek beesési szögét és a rádiójelet sugárzó objektum távolságát tudják meghatározni minden periódusban (mérési ciklusban). Egy ilyen mérési sorozatot melyen a periodikusan sugárzott rádióimpulzusok egyetlen impulzusán különböző pozíciókban elhelyezett mérőeszközök méréseit értjük, burst-nek nevezzük. A rádió jelek különböző feldolgozható információkat is közvetítenek, egyrészt egy egyedi azonosítót, ami egyértelműen azonosítja azt az objektumot amiből a rádiójel származik, másrészt egy folyamatosan növekvő sorszámot ami a mérési ciklusokat számolja így azonosítva egy-egy mérési ciklust (burst-öt). Ezt a mérési ciklust azonosító sorszámot burst_id-nek nevezzük. Ha egy mérési ciklusban elegendő mérési adatot összegyűjtünk, a mérési eredményekből a pozíciót kiszámító algoritmus már ki tudja számítani az objektum pozícióját. A klaszterező algoritmusnak azt kell eldöntenie, hogy mikor küldjük tovább a beérkező adatokat. Az algoritmus nem várhatja egyszerűen minden mérési eredmény beérkezését, ennél jóval kifinomultabb megoldásra van szükség. A cél az, hogy az algoritmus döntse el mikor érdemes az összegyűjtött adatokat a pozíció számításhoz továbbítania. Ehhez két egymásnak ellentmondó célnak kell megfelelnie. Az első, hogy a pozíció számító algoritmusnak az összes rendelkezésre álló mérési adatot meg kell kapnia a mérési pontoktól, a lehető legpontosabb pozíciómeghatározás érdekében. Ugyanakkor a rendszernek nem szabad túl sokáig várnia a mérési adatok begyűjtésére, hiszen minden késedelem az adattovábbításban jelentősen rontja a mozgó objektumra meghatározott pozíció relevanciáját. Egy majdnem pontos pozíció jobb, mint egy teljesen pontos pozíció túl későn. Bemutatunk egy a fenti irányelveket célzó integrált célfüggvényt, és a klaszterezési problémát maximalizációs problémaként definiáljuk.

Minden adattovábbítási p_j időponthoz és k mérési helyhez legyen $e_{jk} = 1$, ha a mérési adatot továbbítottuk a pozíció számításhoz egyébként legyen $e_{jk} = 0$.
Legyen

$$r_j = \max_{i=1}^{n}\{Rcpt(x_i)|x_i \in B'_j\}.$$

A következő maximalizálandó célfüggvényt definiáljuk:

$$f = \sum_{j=1}^{b} \sum_{k=1}^{m} e_{jk} w_k - c \sum_{j=1}^{b} (p_j - r_j).$$

Az NWT (No Waiting Time Algorithm) algoritmus minden burst-höz az első adatot küldi tovább a pozíció kiszámításához, amint az megérkezett.

A CWT (Constant Waiting Time) algoritmus minden burst-nél az első adatcsomag megérkezése után vár még d ideig, majd elküldi az összes addig megérkezett adatot a pozíció meghatározónak.

5. Tézis

7. Tétel. *Nincs olyan algoritmus melynek a versenyképességi hányadosa kisebb, mint* $\frac{\sum_{k=1}^{m} w_k}{w_1} \geq m$.

8. Tétel. *Az NWT algoritmus versenyképességi hányadosa* $\frac{\sum_{k=1}^{m} w_k}{w_1}$.

9. Tétel. *Ha egy burst hossza nem lehet d-nél nagyobb és* $\frac{w_1}{c} > d$, *akkor nincs olyan algoritmus a korlátos modellben, melynek kisebb a versenyképességi hányadosa mint* $\min\{\frac{w_1}{w_1 - c \cdot d}, \frac{\sum_{k=1}^{n} w_k}{w_1}\}$.

10. Tétel. *A CWT algoritmus versenyképességi hányadosa* $\frac{w_1}{w_1 - c \cdot d}$.

1. Kiegészítés. *A korlátos modellben ha* $\frac{w_1}{w_1 - c \cdot d} \leq \frac{\sum_{k=1}^{m} w_i}{w_1}$, *akkor CWT versenyképességi hányadosa a modellben lehetséges legkisebb versenyképességi hányados, egyébként NWT versenyképességi hányadosa éri el a lehetséges legkisebb értéket.*

Bemutatunk egy további algoritmust is (VWT), ami a d paraméter értékét a beérkező adatok alapján folyamatosan megbecsüli, továbbá egy mérésen alapuló tapasztalati elemzést, mely az algoritmusok hatékonyságát vizsgálja a tényleges környezet által gyűjtött adatokon.

Az algoritmus a következő szabályokat használja, az adatok elküldési időpontjának meghatározására és az adatok küldésére:

- Ha az aktuális idő $S(j)+W$ és $DS(j)$ még nem lezárt (nincs továbbítva a pozíció számításhoz), akkor az algoritmus lezárja $DS(j)$-t, és elküldi azt a pozíció számító algoritmusnak. Egyébként legyen

$$PD(j) = \max\{0, \min_{a=0}^{MEM-1}\{p_{j-a} - r_{j-a}\} - INC\}$$

a W változó lehetséges legkisebb csökkentési értéke, mely az utolsó MEM burst minimális hosszát és az INC biztonsági időt veszi figyelembe. Ha $PD(j) \geq DEC$, akkor W-t csökkentjük $PD(j)$-vel.

- Ha az algoritmus a burst összes adatát összegyűjtötte $DS(j)$-be, azaz t várakozási idő elég volt ahhoz, hogy a pozíciómeghatározási infrastruktúra összes mérési adatát összevárjuk, akkor az összegyűljtött adatokat $DS(j)$-t továbbítjuk a pozíció meghatározáshoz. Ez csak úgy történhetett, ha $t \leq S(j) + W$. Ilyenkor az előző pont szerinti a W értékének csökkentése is megtörténhet.

Az algoritmus a következő szabályokat használja az x_i adat megérkezésekor:

- Ha x_i a j burst-höz tartozik, t időpontban érkezik és $DS(j)$ nincs lezárva, akkor bővítjük $DS(j)$-t az új adattal és ellenőrizzük, hogy ez van e még a j burst-höz tartozó hiányzó adat az infrastruktúrából.
- Ha $DS(j)$ már lezárt az r_j időpontban W értékét módosítjuk a $t - S(j) + INC$ értékre.

6. Tézis

Egy új, paraméter tanuláson alapuló algoritmust fejlesztettünk az on-line klaszterezési probléma megoldására. A Fraunhofer intézet szimulációs rendszerében, valós inputokon futtatott tesztek alapján megállapítottuk, hogy az új algoritmus jobb hatékonyságú mint a korábbi, lehető legkisebb versenyképességi hányadossal rendelkező algoritmusok, és akkor is jól használható, ha nincs előzetes információnk az inputról.

Hivatkozások

[1] Albers, S. and Schröder, B. An Experimental Study of Online Scheduling Algorithms. *ACM Journal of Experimental Algorithms*, article 3, 2002.

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

[3] A. Borodin, R. El-Yaniv, *Online Computation and Competitive Analysis* Cambridge University Press, 1998.

[4] M. Brugger, T. Christ, F. Kemeth, S. Nagy, M. Schaefer, M.M. Pietrzyk, The FMCW technology-based indoor localization system, *Proceedings of Ubiquitous Positioning Indoor Navigation and Location Based Service (UPINLBS)*, Helsinki, Finnland, 2010 pp. 1–6.

[5] M. Brugger, F. Kemeth, Locating rate adaptation by evaluating movement specific parameters, *Proceedings of 2010 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, Anaheim, USA, 2010 pp. 127–133.

[6] D. R. Dooly, S. A. Goldman, S. D. Scott: On-line analysis of the TCP acknowledgment delay problem. *J. ACM* **48(2)** 243–273, 2001.

[7] Cs. Imreh, T. Németh, *On time lookahead algorithms for the online data acknowledgement problem* in *32nd International Symposium on Mathematical Foundations of Computer Science, LNCS 4708*, Cesky Krumlov, 2007, pp. 288-297.

[8] Cs. Imreh and T. Németh, Parameter learning algorithm for the online data acknowledgment problem, *Optim. Methods Softw.,* **26**, 3 (2011) 397–404.

[9] T. Németh, B. Gyekiczki, Cs. Imreh, Parameter Learning in Lookahead Online Algorithms for Data Acknowledgment, *IEEE International Symposium on Logistics and Industrial Informatics*, 2011

[10] T. Németh and Cs. Imreh, *Parameter learning online algorithm for multiprocessor scheduling with rejection*, Acta Cybernetica 19(1) (2009), pp. 125–133.

[11] T. Németh, S. Nagy, Cs. Imreh Online data clustering algorithms in an RTLS system *Acta Universitatis Sapientiae, Informatica*, **5**, 1 (2013) 5-15.

[12] W. R. Stevens, *TCP/IP Illustrated, Volume I. The protocols*, Addison Wesley, Reading, 1994