Gyakori elemhalmazok kinyerése

Balambér Dávid

Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Számítástudomány szakirány

2011 március 11.

Tartalom

A probléma

Vásárlói szokások

- Gyakran vásárolt termékek $\Rightarrow$ Nagy profit
- Gyakran **együtt** vásárolt termékek $\Rightarrow$ Nagy profit
  - Egyik árát $\downarrow$ (AKCIÓ), másik árát $\uparrow\uparrow$

$\Rightarrow$ érdemes vizsgálni őket

Tartalom

Gyakori elemhalmaz

- **Elemek halmaza**: $\mathcal{I} = \{i_1, i_2, \ldots, i_m\}$
  - termékek
- **Tranzakciók** (bemeneti sorozat): $\mathcal{T} = \langle t_1, t_2, ..., t_n \rangle$, ahol $t_j \subseteq \mathcal{I}$ ($\mathcal{I}$ hatványhalmaza felett értelmezett sorozat)
  - vásárlások (termékhalmazok)
- $I \subseteq \mathcal{I}$ elemhalmaz **fedése**: azon tranzakciók, melyek részhalmaza I
  - egy termékhalmazt tartalmazó vásárlások
- Az I elemhalmaz **támogatottsága**: a fedésének elemszáma, $supp(I)$
  - egy termékhalmazt hány vásárlás tartalmaz
- Az I elemhalmaz **gyakori elemhalmaz**, ha $supp(I) > min\_supp$, ahol min_supp a támogatottsági küszöb
  - egy termékhalmazt legalább min_supp darab vásárlás tartalmaz
- Az I elemhalmaz **gyakorisága**: $freq(I) = \frac{supp(I)}{|\mathcal{T}|}$

Feltételezések

- $\mathcal{I}$ elemein definiálhatunk rendezést: $ABCDEF\ldots$
  - azonosító szerint, pl. vonalkód
- a tranzakciók elemeit rendezve tároljuk:
  $\langle\{A, C, D\}, \{B, C, E\}, \{A, B, C, E\}, \{B, E\}\rangle$
  $\langle ACD, BCE, ABCE, BE\rangle$
  - a vásárlásokkor rendezzük
- az azonos elemszámú elemhalmazok lexikografikusan rendezhetők:
  $\langle ACD, ABD, ADE, ACE\rangle$
  $\langle ABD, ACD, ACE, ADE\rangle$

Tartalom

Adattárolási módok

- Horizontális: a tranzakciókhoz tároljuk az elemek listáját

		ABCD
1	*AB*	1100
2	*BD*	0101
3	*BCD*	0111
4	*ABCD*	1111

- Vertikális: az elemekhez tároljuk a tranzakciók listáját

	1234	
A	14	1001
B	1234	1111
C	34	0011
D	234	0111

Adattárolási módok

- Relációs: rögzített számú attribútum, több a többhöz kapcsolat

R_{TE}:

T	E
1	A
1	B
2	B
2	D
3	A
3	B
3	C
4	A
4	B
4	C
4	D

- Elemek (termékek) adatai:

R_E:

E_{ID}	Név	...
A	...	...
B	...	...
C	...	...
D	...	...

- Tranzakciók (vásárlások) adatai:

R_T:

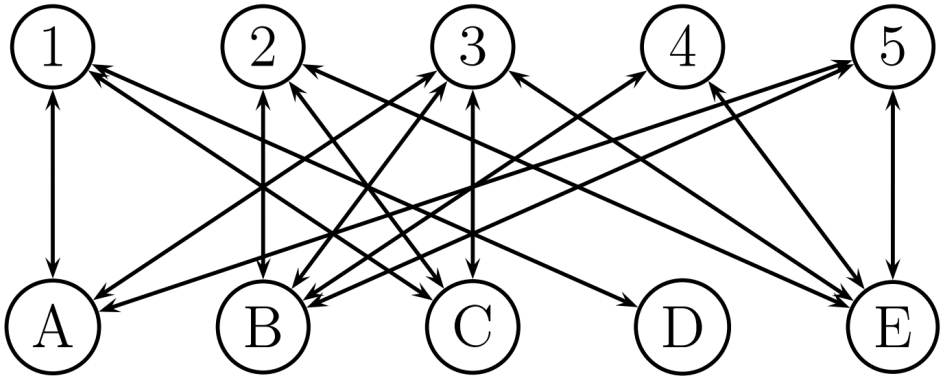
T_{ID}	Idő	...
1	...	...
2	...	...
3	...	...
4	...	...

Ábrázolási módok

- Bináris mátrix:

	ABCD
1	1100
2	0101
3	0111
4	1111

- Páros gráf, $G = (I, T, R)$:



Tartalom

Egyszerű algoritmusok

- Nagyságrendek a gyakorlatban:
  - Elemek száma $|\mathcal{I}| = m$, akár $10^5 - 10^6$
  - Tranzakciók száma $|\mathcal{T}| = n$, akár $10^9 - 10^{10}$
  - gyakori elemhalmazok mérete $\ll |\mathcal{I}|$
  - $|t_j| \ll |\mathcal{I}|$
- Naiv módszer:
  - Határozzuk meg minden elemhalmaz támogatottságát
  - $O(2^m)$ féle elemhalmaz
  - mindegyikre végig kell nézni az összes tranzakciót

Egy konkrét egyszerű algoritmus

- Végigolvassuk a tranzakciókat
- Minden tranzakcióra
  - létrehozunk az összes részhalmazához egy-egy számlálót
  - ha valamely részhalmazra már van számláló, akkor azt növeljük
  - a számláló a támogatottságot jelzi
- Az algoritmus lépésszáma
  - IO szempontjából optimális
  - nagy elemszámú tranzakcióra túl sok számláló kellene: $2^{|t_j|}$ darab
- Javítások
  - csak bizonyos méretű elemhalmazok vizsgálata (nem teljes)
  - számlálók szófában tárolása (számláló elérés $O(|t_j|)$)

Általános algoritmusok felépítése

- A három lépés
  - jelöltek állítása: $J = \{j\}$
  - támogatottságuk meghatározása: $supp(j)$
  - gyakoriak kiválasztása: $GY = \{j \mid supp(j) \geq min\_supp\}$
- A három lépés ismétlődik
- A jelölt-állítás ismétlés nélküli
- A különbség az algoritmusok között:
  - a jelöltek előállításának módja
  - a támogatottság meghatározásának módja
- A tranzakciók elemei rendezetten állnak rendelkezésre
  - pl. minden vásárlásnál rendezzük

Tartalom

Apriori algoritmus

- Iterációk: egyre nagyobb méretű elemhalmazok vizsgálata
  - Gyakori elemhalmaz minden részhalmaza gyakori
  - Csak az olyan elemhalmaz lehet gyakori, melynek részhalmazai gyakoriak
  - $\Rightarrow$ Legyenek azok jelöltek, amelyek minden ismert részhalmaza gyakori
- Kezdetben $l = 0$, $J_l = \{\emptyset\}$

while $|J_\ell| \neq 0$ **do**

$\text{támogatottság\_meghatározás}(\ \mathcal{T}, J_\ell\ );$

$GY_\ell \leftarrow \text{gyakoriak\_kiválogatása}(\ J_\ell, min\_supp\ );$

$J_{\ell+1} \leftarrow \text{jelölt\_előállítás}(\ GY_\ell\ );$

$\ell \leftarrow \ell + 1;$

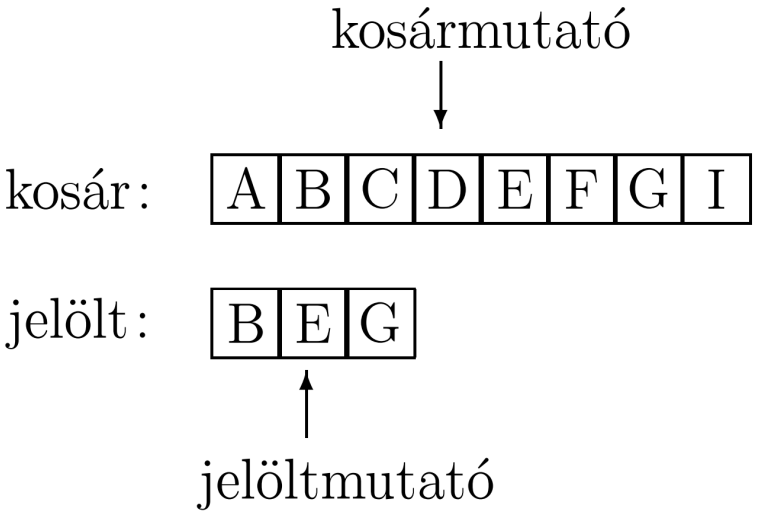
end while

Jelöltek előállítása

- Az l-elemű I_1, I_2 halmazokból $l+1$ elemű előállítása, ha:
  - $I_1 < I_2$ (lexikografikus rendezés szerint)
  - I_1 és I_2 csak utolsó tagjukban térnek el egymástól
- $I_1 \cup I_2$ jelölt lesz, ha minden valódi részhalmaza gyakori
  - ha minden l elemű részhalmaz gyakori, akkor minden valódi részhalmaza is
- Pl. $GY_3 = (ABC, ABD, ACD, ACE, BCD)$
  - $ABCD$ jelölt lesz
  - $ACDE$ nem lesz jelölt
- Csak a jelöltek lehetnek gyakoriak
  - ugyanis ahhoz hogy egy $l+1$ elemű elemhalmaz gyakori lehessen, az összes l elemű részhalmazoknak gyakoriaknak kell lenniük
  - ekkor viszont az $l+1$ elemű elemhalmaz előáll a fenti két szabály betartásával, azaz jelölt lesz

Támogatottság meghatározása

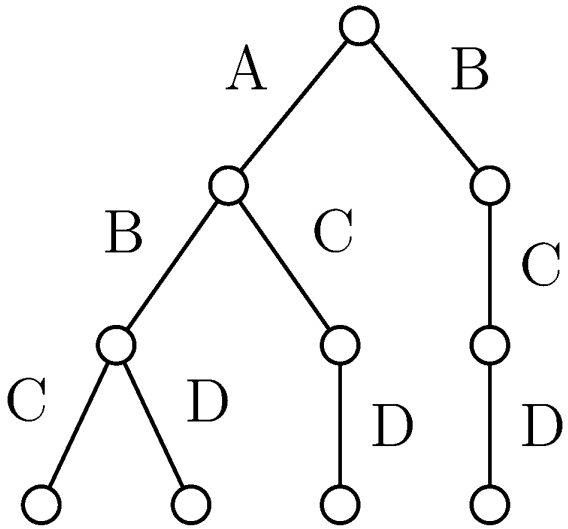
- A tranzakciókat mindenképp végig kell olvasni
- Egyelemű jelöltek esetén:
  - Egy tömbben tároljuk az egyes elemek támogatottságát
- Kételemű jelöltek esetén:
  - Kétdimenziós tömbben tároljuk az egyes elemek támogatottságát
- Általános esetben:



  - Sok jelölt esetén lassú: minden jelöltre külön kell végignézni a tranzakciókat

Támogatottság meghatározása szófával

- A jelöltek támogatottságát szófában tároljuk
- Minden tranzakció beolvasásánál végigmegyünk a szófa megfelelő részein



  - a tranzakciókon csak egyszer megyünk végig
  - a gyakorlatban gyorsabb a jelöltek szófában tárolása
- A támogatottság-meghatározás után töröljük a *min_supp*-nál kisebb támogatottságú leveleket

Jelöltállítás szófával

- A szófa prefix $\Rightarrow$ a közös ősű levelek uniója megfelelő jelölt ha teljesül, hogy a részhalmazai gyakoriak (ezt le kell ellenőrizni)
- Tehát az algoritmus egy szófát épít, egy iteráció lépései:
  - Jelöltek támogatottságának meghatározása
  - Nem gyakori levelek törlése
  - Új jelöltek létrehozása közös ősű jelöltekből
- Nem gyakori jelöltek törlése – gyorsítási lehetőség:
  - Jelöltek támogatottságának meghatározása
  - Új jelöltek létrehozása közös ősű **gyakori** jelöltekből
    - ★ a két lépés összevonható egy szófa-bejárásra

További szófa gyorsítási lehetőségek

- A lexikografikus rendezés választása
  - Egy szófa memóriaigényének minimalizálása a rendezés megfelelő választásával NP-nehéz probléma
  - A gyakorlati tapasztalatok alapján az elemgyakoriság szerint növekvő sorrendet érdemes választani, ennek előnyei:
    - ★ a szófa pontjaiból kevesebb él indul ki
    - ★ a gyökérhez közelebb vannak a ritka elemek
  - A ritka elemekkel kevesebb tranzakció fog egyezni, $\Rightarrow$ a szófa kisebb részét járjuk be $\Rightarrow$ gyorsabb
- Zsákutca nyesés
  - Olyan csúcsok (részfák) törlése amelyek egyik gyereke se gyakori
  - Jelöltállításkor ha preorder járjuk be a szófát, akkor nyugodtan törölhetjük őket

További Apriori gyorsítási lehetőségek

- Ritka elemek szűrése
  - a tranzakciók első végigolvasáskor megállapítjuk a gyakori elemeket
  - majd a nem gyakori elemeket törölhetjük a tranzakciókból
- Túl kis elemszámú tranzakciók törlése
  - az l-edik iterációnál törölhetjük az l-nél nem nagyobb elemszámú tranzakciókat
- Töröljük a tranzakciót ha nem tartalmaz jelöltet
  - Ha nincs benne jelölt, akkor nem tartalmazhat már nagyobb gyakori elemhalmazt
- A tranzakcióban levő jelölteken kívüli elemeket töröljük
  - Pl. $J_3 = \{ABC, ABD, BCD, FGH\}$ esetén
    - ★ $t = ABCDH \rightarrow ABCD$
    - ★ $t = ABCGH \rightarrow ABC \rightarrow \emptyset$

Tartalom

Az Eclat és az FP-Growth algoritmus

- Az Eclat rekurzív, mélységi jellegű bejárást valósít meg
  - mindig egyetlen l méretű jelöltet állít elő két $l-1$ méretűből
  - és rögtön meghatározza a támogatottságát
  - *TID*-halmaz:
    - Pl. $\langle AD, AC, ABCD, B, AD, ABD, D\rangle$ bemenetre $TID(AC) = (2,3)$
    - egy l elemhalmaz *TID* halmazának mérete megadja l támogatottságát
    - egy jelölt *TID*-halmaza megkapható a generátorainak *TID*-halmazának metszeteként:
      $TID(ACD) = TID(AC) \cap TID(AD) = (2,3) \cap (1,3,5,6) = (3)$
- Az FP-Growth szintén rekurzív, mélységi bejárást használ
  - Elemhalmaz vetítése:
    - Pl. $\langle ACD, BCE, ABCE, BE\rangle \mid B = \langle CE, ACE, E\rangle$
  - Először kiszűri a nem gyakori elemeket
  - A szűrt tranzakciókat vetíti gyakori elemekre
  - A vetített tranzakciókra rekurzív hívást végez