Alkalmazott Matematikai Lapok **27** (2010), 107-122.

ZART: EGY MULTIFUNKCIONÁLIS MINTAKERESŐ ALGORITMUS

SZATHMÁRY LÁSZLÓ ÉS BOGNÁR KATALIN

Az adatbányászatban az asszociációs szabályok feltárására számos algoritmust ismerünk. Ebben a cikkben a *Zart* nevű multifunkcionális mintakereső algoritmust mutatjuk be, mely a *Pascal*-algoritmuson alapszik. A *Zart* algoritmus számos olyan műveletet is elvégez, melyek általában egymástól függetlenek, mint például gyakori zárt minták keresése, ill. generátorok hozzárendelése a megfelelő lezártakhoz. A *Zart* ezáltal egy komplett algoritmus a minták ekvivalenciaosztályainak a kiszámítására, melyekben a generátorok és a zárt minták is szerepelnek. Ezen jellemzők miatt a *Zart*-tal könnyen előállíthatók az ún. minimális nem redundáns asszociációs szabályok, melyek az összes asszociációs szabály hasznos és veszteségmentes reprezentációját alkotják. Ezen túlmenően a *Zart* – köszönhetően annak, hogy a *Pascal* kiterjesztése – elég hatékonyan működik mind gyengén, mind pedig erősen korrelált adathalmazokon. A *Zart* algoritmus kiemelt helyen szerepel a CORON rendszerben, mely egy többcélú adatbányász platform.

1. Bevezetés

Az adatbányászatban az asszociációs szabályok feltárása az egyik legfontosabb feladat. Nagyon sok esetben azonban a gyakori mintákból igen nagy számú érvényes asszociációs szabályt lehet előállítani, erősen megnehezítve ezáltal a szabályok felhasználását egy valódi alkalmazásban. Ezen probléma megoldására megszülettek a szabályok különböző tömörített reprezentációi, mint pl. a generikus bázis ($\mathcal{GB}$) [1], az informatív bázis ($\mathcal{IB}$) [1], a reprezentatív szabályok [2], a Duquennes–Guigues-bázis [3], a Luxenburger-bázis [4], stb. Egy nagyon jó összehasonlító elemzés található ezen bázisokról Kryszkiewicz cikkében [5], amelyben a szerző megfogalmazza azokat a feltételeket, melyeknek egy szabály-reprezentációnak meg kell felelnie: legyen *veszteségmentes* (vagyis valamennyi érvényes szabályt vissza lehessen állítani), *helyes* (azaz érvénytelen szabályt ne lehessen belőle előállítani), ill. *informatív* (vagyis bizonyos paramétereket – mint pl. *support* vagy *confidence* értékek – meg lehessen állapítani).

Kryszkiewicz megmutatta, hogy a minimális nem redundáns asszociációs szabályok[1] ($\mathcal{MNR}$) a *cover* operátorral valamennyi érvényes asszociációs szabály veszteségmentes, helyes, ill. informatív reprezentációját alkotják [5]. Ugyanez érvényes a tranzitív kapcsolatokat nem tartalmazó redukált minimális nem redundáns asszociációs szabályokra ($\mathcal{RMNR}$) is, ha a *cover* operátort kiegészítjük a *confidence tranzitivitás* jellemzővel. Az $\mathcal{MNR}$ és $\mathcal{RMNR}$ definícióiból látható, hogy

[1] Ezen szabályokat a 2. részben definiáljuk.

ezen szabályok a gyakori zárt minták, ill. a zárt mintákhoz rendelt generátorok segítségével előállíthatók. A gyakori mintáknak számos tömörített reprezentációja létezik, mint pl. a zárt minták [6, 7, 8], a generátor reprezentáció [9], a diszjunkciómentes minták [10], a diszjunkciómentes generátorok [9], az általánosított diszjunkciómentes generátorok [11] stb. Ezen reprezentációk közül a gyakori zárt mintákat, ill. a gyakori generátorokat felhasználva lehetőségünk nyílik az asszociációs szabályoknak egy olyan tömörített halmazát definiálni, mely veszteségmentes, helyes, ill. informatív is egyben [5]. Ezt a szabályhalmazt minimális nem redundáns szabályoknak ($\mathcal{MNR}$) hívjuk [1]. Ez a halmaz – méretét tekintve – általában nem a legkisebb, viszont jó kompromisszumot nyújt a szabályok száma, ill. a szabályok előállításához szükséges idő között [12].

A [13]-as cikkben Bastide *et al.* bemutatta a *Pascal*-algoritmust, s azt állította, hogy ezen algoritmussal előállíthatók az $\mathcal{MNR}$ szabályok. Sajnos a *Pascal*-algoritmus kimenetéből még nem lehet közvetlenül előállítani ezen szabályokat. Először is szükségünk van a gyakori zárt mintákra is. Másodszor, a gyakori generátorokat hozzá kell rendelni a lezártjaikhoz. A *Zart* algoritmus, mely a *Pascal* kiterjesztése, pontosan ezeket a kiegészítő műveleteket végzi el. Az algoritmus neve amúgy a magyar „zárt" szóból származik, ezzel is utalva arra, hogy az algoritmus a gyakori zárt mintákat is megkeresi. A *Zart* algoritmus kimenetéből már könnyedén előállíthatók az $\mathcal{MNR}$ szabályok. A *Pascal* helyett másik algoritmust is vehettünk volna alapul. A választásunk mégis azért esett a *Pascal*-ra, mert a szintenkénti elven működő gyakori mintákat kereső algoritmusok közül talán ez a leghatékonyabb. Ez a jó teljesítmény annak köszönhető, hogy a mintákat számláló következtető mechanizmus jelentős mértékben csökkenti a költséges adathalmaz-újraolvasások számát.

A *Zart* algoritmust a CORON platformon belül implementáltuk [14].[2] A CORON egy többféle kutatási területen alkalmazható, platformfüggetlen, többcélú adatbányászati eszköztár, amely nem csupán számos adatbányászati algoritmust fog össze, de többféle kiegészítő szolgáltatást is nyújt, mint pl. az adatok előkészítése, tisztítása, ill. az eredmények megjelenítése, értelmezése. Legjobb tudomásunk szerint nem létezik még egy olyan, a CORON-hoz hasonló adatbányászati szoftver, mely direkt módon mintakeresésre, ill. asszociációs szabályok előállítására lett volna kifejlesztve. Cikkünk egy korábban angolul megjelent közleményünk [15] kibővített változata.

Ezen cikk a következőképpen épül fel. A 2. részben áttekintjük az alapvető fogalmakat és definíciókat. A 3. részben a *Zart* algoritmus három fő jellemzőjét mutatjuk be. A 4. részben a *Zart* algoritmust ismertetjük, melynek működését egy konkrét példán keresztül is szemléltetjük. Az 5. részben az $\mathcal{MNR}$ szabályok előállítását részletezzük. A 6. részben a *Zart* algoritmus futási eredményét hasonlítjuk össze a *Pascal*- és *Apriori* algoritmusokkal. Végül a 7. részben a konklúziókkal zárjuk a cikket.

[2] http://coron.loria.fr

2. Fogalmak és definíciók

Vegyük a következő 5×5-ös méretű adathalmazt: $\mathcal{D} = \{(1,\ abde),\ (2,\ ac),\ (3,\ abce),\ (4,\ bce),\ (5,\ abce)\}$ (lásd még 1. táblázat). A cikk további részében erre a példára úgy fogunk utalni mint **„$\mathcal{D}$ adathalmaz"**.

1. táblázat. A példákhoz használt $\mathcal{D}$ adathalmaz.

	a	b	c	d	e
1	x	x		x	x
2	x		x		
3	x	x	x		x
4		x	x		x
5	x	x	x		x

Gyakori minták. Legyen $\mathcal{O} = \{o_1, o_2, \ldots, o_m\}$ objektumok (vagy tranzakciók) egy halmaza, $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$ attribútumoknak egy halmaza, $\mathcal{R} \subseteq \mathcal{O} \times \mathcal{A}$ pedig egy bináris reláció. Az $\mathcal{A}$ halmaz egy részhalmazát *mintának*, egy k darab attribútumot tartalmazó mintát pedig k-hosszúságú mintának nevezünk. Minden tranzakció rendelkezik egy egyedi azonosítóval (*tid*), s a tranzakciók egy halmazát *tidset*nek hívjuk. Egy X mintát tartalmazó tidsetet az X minta *képének* is nevezzük, s $t(X)$-szel jelöljük. Pl. a $\mathcal{D}$ adathalmazban az ab minta képe 135, azaz $t(ab) = 135$.[3] Egy X minta *support* értéke – jelölése $supp(X)$ – az X minta képének a mérete, azaz $supp(X) = |t(X)|$. Vagyis a support azt mutatja meg, hogy egy minta hány tranzakcióban van jelen. Egy X mintát gyakorinak hívunk, ha a support értéke nem kisebb, mint egy adott *minimum support* küszöbérték (jelölése min_supp), azaz $supp(X) \geq min\_supp$.

Ekvivalenciaosztályok. Két minta $X, Z \subseteq \mathcal{A}$ ekvivalens ($X \cong Z$), ha a képük megegyezik, vagyis $t(X) = t(Z)$. Egy P mintával ekvivalens minták halmazát a P minta ekvivalenciaosztályának nevezzük: $[P] = \{Q \subseteq \mathcal{A} \mid P \cong Q\}$. Egy ekvivalenciaosztálynak egyetlen maximális eleme van (zárt minta), s több minimális eleme is lehet (generátorok[4]) [13].

2.1. Definíció. Egy minta *zárt*, ha nincs vele azonos support értékű valódi szuperhalmaza.

[3] A halmazok leírására egyszerűsített jelölést alkalmazunk. Pl. $\{a, b, e\}$ helyet abe-t, $\{1, 3, 5\}$ helyett pedig 135-öt fogunk írni.

[4] A szakirodalomban ezen mintákat többféleképpen is nevezik: kulcsminták, minimális generátorok, szabad minták, kulcsgenerátorok stb.

2.2. Definíció. Egy minta (minimális) *generátor*, ha nincs vele azonos support értékű valódi részhalmaza.

A *lezárási* operátor egy X mintához az X ekvivalenciaosztályában található maximális elemet rendeli hozzá. A hozzárendelt elemet X *lezártjának* nevezzük, s $\gamma(X)$-szel jelöljük.

1. Példa. A $\mathcal{D}$ adathalmazban zárt minta pl. az a, ac és $abce$. Az abc minta (support értéke 2) nem zárt az $abce$ minta miatt (melynek support értéke szintén 2). Az abc minta generátor, ui. valamennyi részhalmaza magasabb support értékkel rendelkezik. A b és abc minták generátorok, melyek lezártjai: $\gamma(b) = be$ és $\gamma(abc) = abce$.

Mintaszámláló következtetés. A support értékre és a generátorokra vonatkozó most következő talajdonságok a [13]-as cikkből származnak:

1. Tulajdonság. Legyen P és Q egy-egy minta.

(i) $P \cong Q \Rightarrow supp(P) = supp(Q)$,

(ii) $P \subseteq Q$ és $(supp(P) = supp(Q)) \Rightarrow P \cong Q$.

2. Tulajdonság. Egy gyakori generátor valamennyi részhalmaza gyakori generátor.

3. Tulajdonság. Egy P minta a.cs.a. generátor, ha $supp(P) \neq min_{p \in P}(supp(P \setminus \{p\}))$.

Az utolsó jellemző szerint egy P minta a.cs.a. generátor, ha P support értéke különbözik a nála eggyel kisebb méretű részhalmazainak a support értékétől. Definíció szerint ui. egy generátornak nem lehet vele azonos support értékű valódi részhalmaza.

Gyakori asszociációs szabályok. Egy asszociációs szabály egy $I_1 \rightarrow I_2$ alakú kifejezés, ahol I_1 és I_2 tetszőleges minták ($I_1, I_2 \subseteq \mathcal{A}$), $I_1 \cap I_2 = \emptyset$ és $I_2 \neq \emptyset$. Egy r: $I_1 \rightarrow I_2$ asszociációs szabály support értéke a következőképpen definiálható: $supp(r) = supp(I_1 \cup I_2)$. Egy r asszociációs szabály *confidence* értéke azt mutatja meg, hogy mi annak a valószínűsége, hogy egy objektum rendelkezik az I_2 mintával, feltéve hogy rendelkezik az I_1 mintával: $conf(r) = supp(I_1 \cup I_2)/supp(I_1)$. Egy r asszociációs szabályt melynek confidence értéke 100%, pontos szabálynak (vagy implikációnak), különben pedig megközelítő szabálynak nevezünk. A supporthoz hasonlóan a confidence-re is megadható egy határérték (*minimum confidence*). Egy r asszociációs szabály érvényes, ha $supp(r) \geq min\_supp$ és $conf(r) \geq min\_conf$. Az érvényes asszociációs szabályok halmazát $\mathcal{AR}$-rel jelöljük.

2. Példa. Nézzünk néhány asszociációs szabályt, melyeket a $\mathcal{D}$ adathalmazból nyerhetünk ki. A $b \rightarrow e$ support értéke 4 (80%), confidence értéke 1 (100%). A $c \rightarrow abe$ support értéke 2 (40%), confidence értéke 0,5 (50%).

Most pedig tekintsük át néhány, asszociációs szabályokat reprezentáló bázis definícióját [16].

2.3. Definíció. Jelölje FCI a gyakori zárt minták halmazát. Egy f gyakori zárt minta esetén jelölje FG_f az f ekvivalenciaosztályában lévő generátorokat. A pontos asszociációs szabályok (implikációk) generikus bázisa:

$$\mathcal{GB} = \{r : g \Rightarrow (f \backslash g) \mid f \in FCI \wedge g \in FG_f \wedge g \neq f\}.$$

2.4. Definíció. Jelölje FCI a gyakori zárt minták, FG pedig a gyakori generátorok halmazát. A megközelítő asszociációs szabályok informatív bázisa:

$$\mathcal{IB} = \{r : g \rightarrow (f \backslash g) \mid f \in FCI \wedge g \in FG \wedge \gamma(g) \subset f\}.$$

2.5. Definíció. Jelölje FCI a gyakori zárt minták halmazát, $\mathcal{IB}$ pedig az előbb definiált megközelítő asszociációs szabályok informatív bázisát. Az $\mathcal{IB}$ tranzitív kapcsolatoktól mentes változatát redukált informatív bázisnak nevezzük:

$$\mathcal{RIB} = \{r : g \rightarrow (f \backslash g) \in \mathcal{IB} \mid \gamma(g) \text{ az } f \text{ legnagyobb valódi részhalmaza az FCI-ben}\}.$$

2.6. Definíció. A minimális nem redundáns asszociációs szabályokat a $\mathcal{GB}$ és $\mathcal{IB}$ halmazok uniója adja ($\mathcal{MNR} = \mathcal{GB} \cup \mathcal{IB}$). Az $\mathcal{MNR}$ tranzitív kapcsolatoktól mentes változatát a $\mathcal{GB}$ és $\mathcal{RIB}$ halmazok uniójaként kapjuk ($\mathcal{RMNR} = \mathcal{GB} \cup \mathcal{RIB}$).

3. Példa. Nézzünk néhány asszociációs szabályt a fenti bázisokból: $\{b \rightarrow e, ab \rightarrow e, ce \rightarrow b\} \in \mathcal{GB}$, $\{b \rightarrow ce, b \rightarrow ace\} \in \mathcal{IB}$, $\{b \rightarrow ce, b \rightarrow ae, bc \rightarrow ae\} \in \mathcal{RIB}$. Az 5. rész végén további példák is találhatók.

3. A Zart fő jellemzői

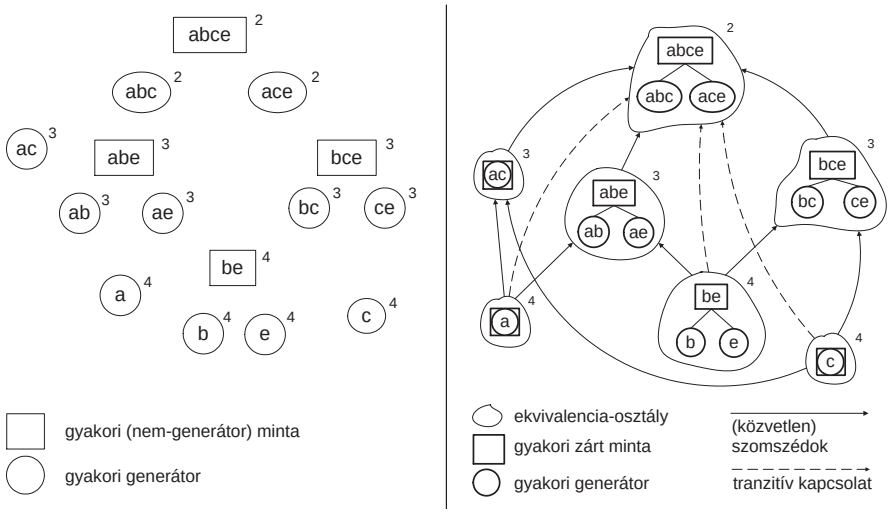
A *Zart* algoritmus három fázisból tevődik össze: **(1)** mintaszámláló következtetés, **(2)** gyakori zárt minták azonosítása, **(3)** a gyakori zárt minták generátorainak a megállapítása.

3.1. Mintaszámláló következtetés

A *Zart* első része a *Pascal*-algoritmus mintaszámláló következtetésére épül. A gyakori minták szintenkénti keresése esetén először az ekvivalenciaosztályok legkisebb elemeit találjuk meg, s ezek a minták pontosan a generátorok. Később, mikor egy nagyobb mintát találunk, leteszteljük, hogy egy már felfedezett ekvivalenciaosztályhoz tartozik-e. Ha igen, akkor nem szükséges újra végigfutni az adathalmazon a minta support értékének megállapításához. Vagyis ezáltal a költséges adathalmaz-újraolvasásokat és support számításokat le lehet csökkenteni csupán a

generátorok esetére. Egy bizonyos szinttől viszont már meglesz az összes generátor, tehát a hátralévő gyakori minták, ill. ezek support értékei megállapíthatóak anélkül, hogy újra végig kellene olvasni az adathalmazt.

Az 1. ábrán (bal oldal) a *Pascal* kimenete látható: az algoritmus megtalálja a gyakori mintákat és megjelöli a gyakori generátorokat. Ha megnézzük az $\mathcal{MNR}$ és $\mathcal{RMNR}$ szabályok definícióit, akkor egyértelmű, hogy ez a kimenet nem elegendő. A *Zart* kimenete szintén az 1. ábrán szerepel (jobb oldal). Mint látható, az algoritmus feltárja az ekvivalenciaosztályokat. Az ábrán csupán az ekvivalenciaosztályok maximális (gyakori zárt minta) és minimális (gyakori generátorok) elemei vannak feltüntetve. A support értékek az osztályok jobb felső sarkában szerepelnek. A *Zart* által produkált kimenet szükséges és elegendő a $\mathcal{GB}$, $\mathcal{IB}$, $\mathcal{RIB}$, $\mathcal{MNR}$ és $\mathcal{RMNR}$ szabályok generálásához.



1. ábra. A *Pascal* (bal oldal) és a *Zart* (jobb oldal) eredménye a $\mathcal{D}$ adathalmazon $min\_supp = 2$ (40%) küszöbérték mellett.

3.2. Zárt minták azonosítása a gyakori minták között

A *Zart* algoritmus második része a zárt mintákat azonosítja a gyakori minták között. Ez az ötlet eredetileg az *Apriori-Close* algoritmusban szerepelt [16]. Definíció szerint egy zárt mintának nincs vele azonos support értékű szuperhalmaza. A k. iteráció során valamennyi k-hosszúságú mintát „zárt"-nak minősítünk. A $(k+1)$. iterációban minden egyes $(k+1)$-hosszúságú minta esetén megnézzük, hogy az adott minta tartalmaz-e vele azonos support értékű k-hosszúságú mintát. Ha igen, akkor a k-hosszúságú minta nem zárt, hiszen van egy vele azonos support

értékű szuperhalmaza. Így a k-hosszúságú minta státusza „nem zárt”-ra változik. Mikor az algoritmus befejezi a futását, a még mindig „zárt”-nak jelölt minták lesznek a tényleges gyakori zárt minták.

3.3. Generátorok és lezártjaik összerendelése

A szintenkénti bejárás következtében mikor megtalálunk egy zárt mintát, akkorra már valamennyi gyakori részhalmazát feltártuk. Ez azt jelenti, hogy már a generátorait is érintettük, csupán be kell őket azonosítani. Megmutatjuk, hogy a generátorok keresési tere leszűkíthető a nem zárt mintákra. Ez a következő két tulajdonságon alapszik:

4. Tulajdonság. Egy zárt minta nem lehet egy nagyobb minta generátora.

5. Tulajdonság. Egy gyakori nem zárt g generátor lezártja nem más, mint a g legkisebb valódi szuperhalmaza a gyakori zárt minták között.

Ezen két tulajdonságot felhasználva a generátorokat a következőképpen találhatjuk meg hatékonyan. A generátorokat egy l listában tároljuk. A k. iteráció során a k-hosszúságú gyakori zárt mintákat kiszűrjük. Minden k-hosszúságú gyakori zárt minta esetén (nevezzük a mintát z-nek) a következő lépéseket kell elvégezni: keressük meg z részhalmazait az l listában, regisztráljuk őket mint z generátorait, majd töröljük őket l-ből. Mielőtt továbblépnénk a $(k+1)$. iterációra, adjuk hozzá az l listához a k-hosszúságú nem zárt generátorokat. A 4-es és 5-ös tulajdonságok garantálni fogják, hogy mikor egy gyakori zárt minta részhalmazait keressük az l listában, akkor csakis a generátorait fogjuk megtalálni. A listából visszaadott minták support értéke azonos a gyakori zárt minta supportjával, így ezt nem kell külön leellenőrizni. Mivel a listában csupán a generátorokat tároljuk, így jóval kevesebb elemet kell tesztelni, mintha az összes gyakori mintát letárolnánk. Mivel a k. lépés során az l listában tárolt legnagyobb minta mérete legfeljebb $(k-1)$ lehet, ezért nem fogjuk megtalálni azon generátorokat, melyek azonosak a lezártjukkal. Ha egy gyakori zárt mintának az algoritmus lefutása után nincs beregisztrált generátora, akkor ez egyszerűen azt jelenti, hogy a minta generátora önmaga. Ami az implementációt illeti, a generátorok tárolására egy hagyományos lista helyett a prefix fa (*trie*) adatstruktúrát javasoljuk, mivel ezen adatstruktúrával nagyon gyorsan meg lehet keresni egy minta részhalmazait.

2. táblázat. A *Zart* által használt táblák és mezők.

C_i	potenciálisan gyakori i-hosszúságú jelöltek mezők: (1) minta, (2) pred_supp, (3) kulcs, (4) support
F_i	gyakori i-hosszúságú minták mezők: (1) minta, (2) kulcs, (3) support, (4) zárt
Z_i	gyakori i-hosszúságú zárt minták mezők: (1) minta, (2) support, (3) gen

minta	–	egy tetszőleges minta
pred_supp	–	az $(i-1)$-hosszúságú gyakori részhalmazok support értékeinek a minimuma
kulcs	–	az adott minta generátor?
zárt	–	az adott minta zárt?
gen	–	az adott zárt minta generátorai

3. táblázat. A *Zart* algoritmus főblokkja.

```
1)  FG ← { };   // globális lista a gyakori generátoroknak
2)  C_1 feltöltése 1-hosszúságú mintákkal;
3)  C_1-ben lévő minták support értékeinek a megállapítasa;
4)  gyakori minták átmásolása C_1-ből F_1-be;
5)  az F_1-ben lévő mintákat jelöljük meg „zárt”-ként;
6)  az F_1-ben lévő mintákat jelöljük meg „kulcs”-ként ha a support
        értékük kisebb, mint az adathalmaz objektumainak a száma;
        // ui. definíció szerint az üres halmaz support értéke 100%
7)  ha a bemeneti adathalmaznak van telített oszlopa, akkor FG ← {∅};
8)  i ← 1;
9)  ciklus
10) {
11)     C_{i+1} ← Zart-Gen(F_i);
12)     ha C_{i+1} üres, akkor lépjünk ki a ciklusból;
13)     állapítsuk meg a „kulcs”-ként megjelölt minták support-ját a C_{i+1}
            táblában;
14)     ha a C_{i+1}-ben van olyan c minta, hogy c.support = c.pred_supp,
            akkor jelöljük meg a c mintát „nem kulcs”-ként;
15)     gyakori minták másolása C_{i+1}-ből F_{i+1}-be;
16)     ha egy F_{i+1}-ben lévő mintának van F_i-beli részhalmaza úgy, hogy
            a két minta support-ja azonos, akkor a részhalmazt jelöljük
            meg „nem zárt”-ként;
17)     a „zárt”-ként megjelölt mintákat másoljuk át F_i-ből Z_i-be;
18)     Find-Generators(Z_i);
19)     i ← i + 1;
20) }
21) minták átmásolása F_i-ből Z_i-be;
22) Find-Generators(Z_i);
```

4. A Zart algoritmus

4.1. Pszeudó kód

A *Zart* algoritmus három különböző táblát használ, ezek leírása az 2. táblázatban látható. Az algoritmus főblokkja a 3. táblázatban szerepel. Az algoritmus részletesebb pszeudó kódja a [14]-es dolgozatban szerepel.

Zart-Gen függvény. A függvény bemenete egy F_i tábla, mely i-hosszúságú gyakori mintákat tartalmaz. A függvény kimenete a C_{i+1} tábla. Az eljárás a következő: a függvény feltölti a C_{i+1} táblát az F_i-ben szereplő minták 1-gyel nagyobb szuperhalmazaival. A C_{i+1} táblában szereplő minták pred_supp értékei a náluk 1-gyel kisebb méretű gyakori részhalmazaik support értékeinek a minimumát fogják felvenni. Ha valamelyik részhalmaz nem generátor, akkor az adott $(i+1)$-hosszúságú minta sem generátor, így a support értéke azonos lesz a pred_supp értékkel.

Find-Generators eljárás. Az eljárás bemenetként kap egy Z_i táblát, majd a táblában található valamennyi z gyakori zárt minta esetén a következő műveleteket végzi el: megkeresi z valódi részhalmazait a globális FG listában, a találatokat beregisztrálja mint z generátorait, kitörli őket az FG listából, majd az F_i-ben található nem zárt generátorokat hozzáadja az FG listához.

4.2. Futási példa

A *Zart* végrehajtását a $\mathcal{D}$ adathalmazon a 4. táblázatban szemléltetjük. Az algoritmus először végigolvassa az adathalmazt az 1-hosszúságú minták support értékeinek megállapítása végett. A d minta törlésre kerül, mivel nem gyakori. A következő iteráció során valamennyi 2-hosszúságú jelöltet előállítjuk, ill. megállapítjuk a support értékeiket. A C_2-ben van egy minta, amelynek a support értéke azonos az egyik részhalmazának a supportjával, így a be minta nem generátor. Az F_2 táblázat segítségével megállapítjuk, hogy az F_1-ben található b és e minták nem zártak, mivel van azonos support-ú szuperhalmazuk. A megmaradt a és c zárt mintákat átmásoljuk a Z_1 táblába, majd megállapítjuk a generátoraikat. A gyakori generátorok globális listájában (FG) – mely még üres – nincs részhalmazuk, ami azt jelenti, hogy az a és c minták generátorok. Az F_1-ben lévő nem zárt generátorokat (b és e) hozzáadjuk az FG listához. A C_3 táblában az abe és bce minták nem generátorok. Az F_3 tábla alapján az ab, ae, bc és ce minták nem zártak. A megmaradt ac és be zárt mintákat átmásoljuk a Z_2 táblába. Az ac generátora önmaga, míg a be generátorai b és e. Ezen két generátort kitöröljük az FG listából, majd az ab, ae, bc és ce mintákat hozzáadjuk az FG-hez. Az $abce$ jelölt szintén nem generátor, s mivel nincs több generátor-jelölt C_4-ben, ezért innentől már nem kell többször végigolvasni az adathalmazt a support értékek megállapításához. Az ötödik iteráció során már nem tudunk új jelöltet előállítani, így az algoritmus kilép a ciklusból. Az $abce$ generátorait kiolvassuk az FG listából.

Az algoritmus tehát megkeresi valamennyi gyakori mintát, gyakori zárt mintát, ill. a zárt minták generátorait is (lásd 5. táblázat, jobb oldal). A táblázatban a

4. táblázat. A *Zart* végrehajtása a $\mathcal{D}$ adathalmazon ($min\_supp = 2$ (40%)).

adathalmaz végigolvasása$_1$ →

C_1	pred_supp	kulcs	supp
a			4
b			4
c			4
d			1
e			4

F_1	kulcs	supp	zárt
a	igen	4	igen
b	igen	4	~~igen~~
c	igen	4	igen
e	igen	4	~~igen~~

Z_1	supp	gen
a	4	
c	4	

$FG_{előtte} = \{\}$
$FG_{utána} = \{b, e\}$

adathalmaz végigolvasása$_2$ →

C_2	pred_supp	kulcs	supp
ab	4	igen	3
ac	4	igen	3
ae	4	igen	3
bc	4	igen	3
be	4	~~igen~~	4
ce	4	igen	3

F_2	kulcs	supp	zárt
ab	igen	3	~~igen~~
ac	igen	3	igen
ae	igen	3	~~igen~~
bc	igen	3	~~igen~~
be	nem	4	igen
ce	igen	3	~~igen~~

Z_2	supp	gen
ac	3	
be	4	$\{b, e\}$

$FG_{előtte} = \{b, e\}$
$FG_{utána} = \{ab, ae, bc, ce\}$

adathalmaz végigolvasása$_3$ →

C_3	pred_supp	kulcs	supp
abc	3	igen	2
abe	3	~~igen~~	3
ace	3	igen	2
bce	3	~~igen~~	3

F_3	kulcs	supp	zárt
abc	igen	2	~~igen~~
abe	nem	3	igen
ace	igen	2	~~igen~~
bce	nem	3	igen

Z_3	supp	gen
abe	3	$\{ab, ae\}$
bce	3	$\{bc, ce\}$

$FG_{előtte} = \{ab, ae, bc, ce\}$
$FG_{utána} = \{abc, ace\}$

C_4	pred_supp	kulcs	supp
$abce$	2	~~igen~~	2

F_4	kulcs	supp	zárt
$abce$	nem	2	igen

Z_4	supp	gen
$abce$	2	$\{abc, ace\}$

$FG_{előtte} = \{abc, ace\}$
$FG_{utána} = \{\}$

C_5	pred_supp	kulcs	supp
$\emptyset$			

zárt mintákat „+” jellel láttuk el. A support értékeket a minták mellett zárójelben tüntettük fel. Ha az algoritmus üresen hagyja egy zárt minta generátorait, akkor ez azt jelenti, hogy a generátor azonos a zárt mintával (lásd az a, c és ac mintákat a 4. táblázatban).

5. Minimális nem redundáns asszociációs szabályok előállítása a Zart algoritmussal

Ha a gyakori mintákból állítjuk elő az összes érvényes asszociációs szabályt, akkor valószínűleg túl sok szabályt fogunk kapni, melyek között számos szabály redundáns. Pl. a $\mathcal{D}$ adathalmazból $min\_supp = 2$ (40%) és $min\_conf = 50\%$ mellett nem kevesebb, mint 50 szabályt tudunk kinyerni. Figyelembe véve az adathalmaz 5×5-ös méretét, ez a mennyiség óriási. Hogyan tudnánk megtalálni a legérdekesebb szabályokat? Hogyan lehetne elkerülni a redundanciát, ill. csökkenteni a szabályok számát? A minimális nem redundáns asszociációs szabályok ($\mathcal{MNR}$) a segítségünkre lehetnek. A 2.6. definíció alapján egy $\mathcal{MNR}$ szabály

5. táblázat. A *Zart* algoritmus kimenete.

Összes gyakori minta ($\bigcup_i F_i$)		Összes gyakori zárt minta és generátoraik ($\bigcup_i Z_i$)
a (4) +	be (4) +	a (4); [a]
b (4)	ce (3)	c (4); [c]
c (4) +	abc (2)	ac (3); [ac]
e (4)	abe (3) +	be (4); [b, e]
ab (3)	ace (2)	abe (3); [ab, ae]
ac (3) +	bce (3) +	bce (3); [bc, ce]
ae (3)	$abce$ (2) +	$abce$ (2); [abc, ace]
bc (3)		

formája a következő: a bal oldalon egy gyakori generátor áll, a bal és jobb oldal uniója egy gyakori zárt minta, s a generátor valódi részhalmaza ezen zárt mintának. Az $\mathcal{MNR}$ szabályok előállítására a gyakori zárt mintákra és ezen minták hozzátársított generátoraira van szükség. Mivel a *Zart* algoritmus mindkét halmazt megkeresi, ezért az algoritmus kimenetéből (lásd 5. táblázat) közvetlenül generálhatók ezek a szabályok. A generátorok valódi zárt szuperhalmazainak a gyors azonosítására érdemes a gyakori zárt mintákat egy prefix fában (*trie*) tárolni.

Az $\mathcal{MNR}$ szabályokat a következőképpen lehet előállítani: minden egyes P_1 gyakori generátornak keressük meg a zárt szuperhalmazait (P_2), majd adjuk hozzá az $\mathcal{MNR}$ halmazhoz az $r : P_1 \to P_2 \setminus P_1$ szabályt. Pl. az e generátor felhasználásával az 1. ábrán (jobb oldal) három szabály generálható. Az ekvivalenciaosztályokon belül előállított szabályok alkotják a generikus bázist ($\mathcal{GB}$), melyek mindegyike pontos asszociációs szabály ($e \Rightarrow b$). A különböző ekvivalenciaosztályok között előállított szabályok mind megközelítő szabályok ($e \to bc$ és $e \to abc$). Az $\mathcal{RMNR}$ szabályok esetén a generátorokhoz csupán azon gyakori zárt szuperhalmazokat keressük meg, ahol a zárt minta ekvivalenciaosztálya közvetlen szomszédja a generátor ekvivalenciaosztályának. Vagyis, maradva az előző példánál, csupán három $\mathcal{RMNR}$ szabályt tudunk előállítani az e generátorral: $e \Rightarrow b$, $e \to bc$ és $e \to ab$. Az 6. táblázatban megfigyelhetők a különböző szabályhalmazok méretei.[5] Mint látható, ritka adathalmazok esetén (pl. T20I6D100K) az $\mathcal{MNR}$ szabályok száma majdnem ugyanannyi, mint az összes lehetséges szabály száma ($\mathcal{AR}$). Viszont sűrű, erősen korrelált adathalmazok esetén (pl. C20D10K vagy MUSHROOMS) a szabályhalmazok méretbeli különbsége már jelentős. Az $\mathcal{RMNR}$ halmaz mérete mindig jóval kisebb, mint az $\mathcal{AR}$ halmaz mérete, mind ritka, mind pedig sűrű adathalmazok esetén.

6. táblázat. Asszociációs szabályhalmazok méretbeli összehasonlítása.

adathalmaz (min_supp)	min_conf	$\mathcal{AR}$ (összes érvényes szabály)	$\mathcal{GB}$	$\mathcal{IB}$	$\mathcal{RIB}$	$\mathcal{MNR}$ ($\mathcal{GB} \cup \mathcal{IB}$)	$\mathcal{RMNR}$ ($\mathcal{GB} \cup \mathcal{RIB}$)
$\mathcal{D}$ (40%)	50%	50	8	17	13	25	21
T20I6D100K (0.5%)	90%	752,715	232	721,716	91,422	721,948	91,654
	70%	986,058		951,340	98,097	951,572	98,329
	50%	1,076,555		1,039,343	101,360	1,039,575	101,592
	30%	1,107,258		1,068,371	102,980	1,068,603	103,212
C20D10K (30%)	90%	140,651	967	8,254	2,784	9,221	3,751
	70%	248,105		18,899	3,682	19,866	4,649
	50%	297,741		24,558	3,789	25,525	4,756
	30%	386,252		30,808	4,073	31,775	5,040
Mushrooms (30%)	90%	20,453	544	952	682	1,496	1,226
	70%	45,147		2,961	1,221	3,505	1,765
	50%	64,179		4,682	1,481	5,226	2,025
	30%	78,888		6,571	1,578	7,115	2,122

6. Teszteredmények

Tesztjeink során a *Zart* algoritmust az *Apriori* és a *Pascal*-algoritmusokkal mértük össze. Mindhárom algoritmust Java-ban implementáltuk a Coron adatbányász platformban [14].[6] A teszteket egy Intel Pentium IV 2.4 GHz-es gépen végeztük el Debian GNU/Linux operációs rendszer alatt. Az adott gép 512 MB RAM-mal rendelkezett. Valamennyi válaszidő valós idő, melyet a Unix rendszereken használt *time* paranccsal mértünk le a bemenet és a kimenet között. A tesztekhez a következő adathalmazokat használtuk fel: T20I6D100K, C20D10K és Mushrooms. A T20[7] egy ritka adathalmaz, s felépítésében a bevásárlóközpontok adatbázisaira hasonlít, melyek általában gyengén korrelált adatokat tartalmaznak. A C20 egy népszámlálás részadatait tartalmazza, míg a Mushrooms különböző gombák jellemzőit írja le. Ez utóbbi két adathalmaz erősen korrelált. Korábbi kutatási eredmények szerint a gyengén korrelált adatok, mint pl. a szintetikus adatok, viszonylag könnyű feladatot jelentenek a gyakori mintákat kereső algoritmusok számára, ui. ezen adathalmazokban kevés a gyakori minta. Az ilyen típusú adatokra valamennyi algoritmus hasonló futási időt produkált. Ezzel szemben viszont a sűrű és erősen korrelált adatok sokkal nagyobb kihívást jelentenek, mely arra vezethető vissza, hogy itt meglehetősen nagy lehet a különbség a gyakori és a gyakori zárt minták száma között. Nagyon sok valós életből vett adathalmaz ilyen tulajdonságokkal rendelkezik. Az algoritmusok által produkált futási eredményeket a 7. táblázatban foglaltuk össze.

[5] Definíció szerint a $\mathcal{GB}$ halmazba tartozó szabályok confidence értéke 100%.
[6] http://coron.loria.fr
[7] http://www.almaden.ibm.com/software/quest/Resources/

7. táblázat. A *Zart* futási ideje, ill. egyéb statisztikák (gyakori minták (GyM-k) száma, gyakori zárt minták (GyZM-k) száma, gyakori generátorok (GyG-ok) száma, GyZM-k és GyM-k aránya, GyG-ok és GyM-k aránya).

min_supp	futási idő (mp.)			# GyM	# GyZM	# GyG	$\frac{\#GyZM}{\#GyM}$	$\frac{\#GyG}{\#GyM}$
	Apriori	Pascal	Zart					
T20I6D100K								
2%	72.67	71.15	71.16	378	378	378	100.00%	100.00%
1%	107.63	106.24	107.69	1,534	1,534	1,534	100.00%	100.00%
0.75%	134.49	132.00	133.00	4,710	4,710	4,710	100.00%	100.00%
0.5%	236.10	228.37	230.17	26,836	26,208	26,305	97.66%	98.02%
0.25%	581.11	562.47	577.69	155,163	149,217	149,447	96.17%	96.32%
C20D10K								
50%	61.18	16.68	17.94	1,823	456	456	25.01%	25.01%
40%	71.60	19.10	19.22	2,175	544	544	25.01%	25.01%
30%	123.57	26.74	26.88	5,319	951	967	17.88%	18.18%
20%	334.87	53.28	54.13	20,239	2,519	2,671	12.45%	13.20%
10%	844.44	110.78	118.09	89,883	8,777	9,331	9.76%	10.38%
MUSHROOMS								
60%	3.10	2.04	2.05	51	19	21	37.25%	41.18%
50%	6.03	3.13	3.13	163	45	53	27.61%	32.52%
40%	13.93	6.00	6.03	505	124	153	24.55%	30.30%
30%	46.18	12.79	12.84	2,587	425	544	16.43%	21.03%
20%	554.95	30.30	34.88	53,337	1,169	1,704	2.19%	3.19%

6.1. Gyengén korrelált adatok

A T20 szintetikus adathalmaz a bevásárlóközpontok adatbázisait utánozza, így egy ritka és gyengén korrelált adathalmazról van szó. Ezen adathalmazban kevés a gyakori minta, s majdnem valamennyi gyakori minta generátor. Az *Apriori*, *Pascal*- és *Zart* algoritmusok hasonlóképpen viselkednek. Mint látható, a T20 adathalmazban 0.75% minimum support felett valamennyi gyakori minta zárt és generátor is egyszerre. Ez azt jelenti, hogy minden egyes ekvivalenciaosztálynak csupán egyetlen eleme van. Emiatt a *Zart* és *Pascal*-algoritmusok nem tudják kihasználni a mintaszámláló következtetés előnyeit, így ugyanúgy kénytelenek dolgozni mint az *Apriori*.

6.2. Erősen korrelált adatok

A C20 és MUSHROOMS adathalmazokban a gyakori generátorok száma sokkal kevesebb, mint az összes gyakori minta száma. Emiatt – köszönhetően a mintaszámláló következtetésnek – a *Zart* algoritmusnak sokkal kevesebb minta support-ját kell megállapítania mint az *Apriori*-nak. Valamennyi esetben megfigyelhető, hogy a *Zart* és *Pascal*-algoritmusok futási ideje csaknem azonos, vagyis a *Zart* algoritmus extra jellemzői (úgymint zárt minták azonosítása és a zárt minták generátorainak a megkeresése) nem okoznak szinte semmiféle teljesítménycsökkenést. Az *Apriori* nagyon hatékony ritka adathalmazokon, erősen korrelált adatok esetében viszont a másik két algoritmus sokkal jobban teljesít.

7. Konklúzió és jövőbeli tervek

Ebben a cikkben a *Zart* nevű multifunkcionális mintakereső algoritmust mutattuk be, mely a *Pascal*-algoritmus kiterjesztése. A *Pascal*-tól eltérően a *Zart* azonosítani tudja a gyakori zárt mintákat, ill. megkeresi a zárt minták generátorait. Megmutattuk, hogy szükség van ezen extra tulajdonságokra a minimális nem redundáns asszociációs szabályok előállításához. A teszteredmények szerint a *Zart* a *Pascal*-lal majdnem azonos válaszidőket produkál mind gyengén, mind pedig erősen korrelált adatokon. Tehát a többletkimenet nem megy a teljesítmény rovására.

Érdekes lenne megvizsgálni azt a kérdést, hogy vajon a *Zart*-ban bemutatott ötlet általánosítható-e. Vajon bármely gyakori mintákat kereső algoritmus kiterjeszthető ilyen formában, legyen az akár szélességi, akár mélységi kereső? Vajon lehetséges lenne úgy kiterjeszteni ezeket az algoritmusokat, hogy ne csak az összes érvényes szabályt, hanem közvetlenül a minimális nem redundáns szabályokat is elő lehessen velük állítani? A jövőben ezen kérdésekre szeretnénk választ találni.

Hivatkozások

[1] Bastide, Y. and Taouil, R. and Pasquier, N. and Stumme, G. and Lakhal, L.: *Mining Minimal Non-Redundant Association Rules Using Frequent Closed Itemsets.* In: Proc. of the 1st Intl. Conf. on Computational Logic (CL '00). Volume 1861 of LNAI., Springer (2000) 972–986.

[2] Kryszkiewicz, M.: *Representative Association Rules.* In: Proc. of the 2nd Pacific- Asia Conf. on Research and Development in Knowledge Discovery and Data Mining (PAKDD '98). Volume 1394 of LNCS., Springer-Verlag (1998) 198–209.

[3] Guigues, J. L., Duquenne, V.: *Familles minimales d'implications informatives résultant d'un tableau de données binaires.* Math. et Sci. Hum. **95** (1986) 5–18.

[4] Luxenburger, M.: *Implications partielles dans un contexte.* Mathématiques, Informatique et Sciences Humaines **113** (1991) 35–55.

[5] Kryszkiewicz, M.: *Concise Representations of Association Rules.* In: Proc. of the ESF Exploratory Workshop on Pattern Detection and Discovery. (2002) 92–109.

[6] Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: *Efficient Mining of Association Rules Using Closed Itemset Lattices.* Inf. Syst. **24**(1) (1999) 25–46.

[7] Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: *Discovering Frequent Closed Itemsets for Association Rules.* LNCS **1540** (1999) 398–416.

[8] Zaki, M. J., Hsiao, C. J.: *CHARM: An Efficient Algorithm for Closed Itemset Mining.* In: Proc. of SDM '02. (2002) 33–43.

[9] Kryszkiewicz, M.: *Concise Representation of Frequent Patterns Based on Disjunction-Free Generators.* In: Proc. of ICDM '01, Washington, DC, USA, IEEE Computer Society (2001) 305–312.

[10] Bykowski, A., Rigotti, C.: *A Condensed Representation to Find Frequent Patterns.* In: Proc. of PODS '01, ACM Press (2001) 267–273.

[11] Kryszkiewicz, M., Gajek, M.: *Why to Apply Generalized Disjunction-Free Generators Representation of Frequent Patterns?* In: Proc. of ISMIS 2002, Lyon, France, Springer-Verlag Berlin / Heidelberg (2002) 383–392.

[12] Pasquier, N.: *Mining Association Rules Using Formal Concept Analysis.* In: Proc. of ICCS '00, Shaker-Verlag (2000) 259–264.

[13] Bastide, Y., Taouil, R., Pasquier, N., Stumme, G., Lakhal, L.: *Mining Frequent Patterns with Counting Inference.* SIGKDD Explor. Newsl. **2**(2) (2000) 66–75.

[14] Szathmary, L.: *Symbolic Data Mining Methods with the Coron Platform.* PhD Thesis in Computer Science, University Henri Poincaré – Nancy 1, France (2006)

[15] Szathmary, L., Napoli, A., Kuznetsov, S. O.: *ZART: A Multifunctional Itemset Mining Algorithm.* In: Proc. of the 5th Intl. Conf. on Concept Lattices and Their Applications (CLA '07), Montpellier, France (2007) 26–37.

[16] Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: *Closed Set Based Discovery of Small Covers for Association Rules.* In: Proc. of BDA '99. (1999) 361–381.

(Beérkezett: 2009. augusztus 28.)

SZATHMÁRY LÁSZLÓ
Dépt. d'Informatique UQAM, C.P. 8888,
Succ. Centre-Ville, Montréal H3C 3P8, Canada
Debreceni Egyetem
Informatikai Kar
Debrecen H-4010 Pf. 12
Szathmary.L@gmail.com

BOGNÁR KATALIN
Debreceni Egyetem
Informatikai Kar
Debrecen H-4010 Pf. 12
bognar.katalin@inf.unideb.hu

ZART: A MULTIFUNCTIONAL ITEMSET MINING ALGORITHM

László Szathmáry and Katalin Bognár

In this paper, we present and detail a multifunctional itemset mining algorithm called *Zart*, which is based on the *Pascal* algorithm. *Zart* shows a number of additional features and performs the following, usually independent, tasks: identify frequent closed itemsets and associate generators to their closures. This makes *Zart* a complete algorithm for computing classes of itemsets

including generators and closed itemsets. These characteristics allow one to extract minimal non-redundant association rules, a useful and lossless representation of association rules. In addition, being based on the *Pascal* algorithm, *Zart* has a rather efficient behavior on weakly and strongly correlated data. Accordingly, *Zart* is at the heart of the CORON platform, which is a domain independent, multi-purposed data mining platform, incorporating a rich collection of data mining algorithms.