



**Pannon Egyetem  
Mérnöki Kar  
Kari Tudományos Diákköri  
Konferencia  
2012.**



**Új hasonlóság mérték alapú megoldások  
gyakori elemhalmazok feltárására, elemzésére  
és vizuális megjelenítésére**

Készítette: Bartos Anikó, Gazdaságinformatikus BSc, IV. évf.

Cseresnyés Adrienn, Gazdaságinformatikus BSc, IV. évf.

Témavezető: Dr. Abonyi János, egyetemi tanár

Pannon Egyetem, Mérnöki Kar, Folyamatmérnöki Intézeti Tanszék

Veszprém

**Tartalomjegyzék**

|                                                                           |    |
|---------------------------------------------------------------------------|----|
| Kivonat .....                                                             | 1  |
| Jelölésjegyzék .....                                                      | 2  |
| 1. Bevezetés és irodalmi áttekintés .....                                 | 3  |
| 2. Gyakori elemhalmazok .....                                             | 7  |
| 3. A Bittable algoritmus .....                                            | 9  |
| 3.1. Mátrixok szorzásán alapuló hatékonyságnövelő technika .....          | 9  |
| 3.2. Mátrixok szorzásán alapuló hatékonyságnövelő technika elemzése ..... | 10 |
| 3.3. Az Apriori algoritmus .....                                          | 11 |
| 4. Bittable vizualizációja .....                                          | 14 |
| 4.1 Biclustering .....                                                    | 14 |
| 4.2 Az adatok megjelenítésének fontossága .....                           | 16 |
| 4.3 Az adatok megjelenítése VAT algoritmussal .....                       | 17 |
| 5. Kísérleti rész .....                                                   | 19 |
| 5.1. A kifejlesztett algoritmusok és program .....                        | 18 |
| 5.2. Illusztratív példa .....                                             | 21 |
| 5.3. Mérnöki alkalmazás .....                                             | 25 |
| 6. Összefoglalás .....                                                    | 27 |
| 7. Irodalomjegyzék .....                                                  | 28 |
| 8. Mellékletek .....                                                      | 30 |
| 8.1. A továbbfejlesztett bittable technika pszeudokódja .....             | 29 |
| 8.2. A különböző elemek kiválogatása .....                                | 32 |
| 8.3. Bináris mátrixba való konvertálás .....                              | 33 |
| 8.4. Bittable technika alkalmazása .....                                  | 34 |
| 8.5. A VAT algoritmus használata .....                                    | 35 |

**Kivonat**

A mérnöki gyakorlatban egyre gyakrabban alkalmazott adatbányászati technikák célja a technológiák üzemeltetése és a termékfejlesztés során keletkező adatokból potenciálisan hasznos, előzetesen nem ismert információk feltárása. A kutatás során olyan jellegű információ feltárását céloztuk meg, amely segítségével a lehető legtágabb értelemben vett eseménysorozatok és az ezekhez kapcsolódó komplex folyamatok hatékonyan jellemezhetőek.

A vásárlói kosár elemzés teljesen újszerű továbbfejlesztéseként nem csupán a termékek és a fogyasztók szimultán elemzését tűztük ki célul, hanem az adatbányászat eredményének értelmezését megkönnyítendő a generált gyakori elemhalmazok szemléletes ábrázolását is. A javasolt megoldás elvi háttereként alkalmazott gyakori elemhalmazok feltárására az irodalomban számos jól alkalmazható célirányos algoritmus található, melyek közül az Apriori az egyik legismertebb és legszélesebb körben alkalmazott. Tudományos munkánk során ennek a könnyen értelmezhető és implementálható algoritmusnak a továbbfejlesztésére tettünk kísérletet.

A gyakori elempárok kinyerésére az elemek leszámlálása helyett egy kevésbé ismert Bittable reprezentációt segítségül hívva alapvető mátrix és vektorműveletekre támaszkodó algoritmus készíthető. A TDK dolgozatunk egyik fontos eredménye ennek a Bittable algoritmusnak a továbbfejlesztése. Felismertük, hogy megfelelő mátrix műveletekkel becslést adhatunk a tranzakciók információtartalmára, mely alapján az algoritmus keresési tere jelentősen csökkenthető, azaz a gyakori elemhalmaz keresés hatékonysága növelhető.

A Bittable, azaz a vásárlói tranzakciókat tartalmazó mátrix megjelenítésére a Visual Assessment of (Cluster) Tendency algoritmus átértelmezésén alapuló megoldásra tettünk javaslatot a feltárt zárt elemhalmazokon számított Tanimoto hasonlóság mérték felhasználásával. A generált megoldások a manapság aktívan kutatott biclustering eszközök fejlesztésének irányába mutatnak. Munkánk eredményeként tehát új algoritmusokat dolgoztunk ki a gyakori elemhalmazok feltárására és vizualizációjára, mely algoritmusokat MATLAB fejlesztői környezetben implementáltuk, és benchmarking valamint folyamatmérnöki problémákon generált eredmények értékelésével teszteltük.

**Jelölésjegyzék**

Az alábbi táblázat tartalmazza a dolgozatban használt legfontosabb jelöléseket.

| <b>Jelölés</b>            | <b>Jelentés</b>                                                                                                                                               |
|---------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| $T = \{T_1, \dots, T_N\}$ | tranzakciók halmaza                                                                                                                                           |
| $N$                       | tranzakciók száma                                                                                                                                             |
| $n$                       | elemek, termékek száma, melyekből a tranzakciók felépülnek                                                                                                    |
| $P = \{P_1, \dots, P_n\}$ | elemek, termékek halmaza                                                                                                                                      |
| <b>B</b>                  | a tranzakciókból és termékekből képzett bináris mátrix                                                                                                        |
| $F = \{F_1, \dots, F_k\}$ | gyakori elemhalmazok, melyek tartalmazzák azon elemeket, melyek közös előfordulása gyakori                                                                    |
| $\sigma$                  | támogatottság (support), gyakoriságküszöb, egy olyan korlát, ami meghatározza, hogy hány %-os tartalmazottságtól számítanak az elemek és elempárok gyakorinak |
| $bmax$                    | a $(B \cdot B')$ mátrixban szereplő legnagyobb érték                                                                                                          |
| $IT\{k\}$                 | érdekes, közös terméket vagy termékeket vásárolt vásárlópárok                                                                                                 |
| $Bitems\{k\}$             | azon termékeket gyűjtjük ide, melyek elérték a megfelelő támogatottságot                                                                                      |
| $Bi\{k\}$                 | a supportnak megfelelő $k$ termék támogatottsága <b>B</b> mátrixból véve                                                                                      |
| $[dum1, dum2]$            | általánosan jelzi több elem indexelését. <i>dum1</i> : sor, <i>dum2</i> : oszlop.                                                                             |

## 1. Bevezetés és irodalmi áttekintés

Az elmúlt évtizedekben az informatika előretörésével igény született a keletkező információk tárolásán túl további, hasznos felhasználásukra. A mérnöki gyakorlatban is egyre gyakrabban alkalmazott adatbányászati technikák célja a technológiák üzemeltetése, és a termékfejlesztés során keletkező adatokból potenciálisan hasznos, előzetesen nem ismert információknak a feltárása (Han, 2004.). Az ilyen adatok kinyeréséhez napjainkban már számos, különböző típusú adatbányászati algoritmus áll rendelkezésre, attól függően, hogy milyen információra vagyunk kíváncsiak.

Az első ilyen adatbányászati módszerek egyike a vásárlói kosár elemzés (Berry - Linoff, 1997.), ahol azt próbálják kiszűrni, melyek azok a termékek, vagy termékpárok, termékhármak, stb. amelyeket viszonylag gyakran vásárolnak együtt a fogyasztók, ezekből pedig következtetéseket lehet levonni, és asszociációs szabályokat (Bodon, 2010.) lehet állítani. Az elemzés során az adott üzleti vásárlást tranzakciónak  $T = \{T_1, \dots, T_N\}$  hívják, a vásárolt termékeket pedig elemeknek vagy termékeknek  $P = \{P_1, \dots, P_n\}$ . A gyakran előforduló termékeket, termékpárokat pedig gyakori elemhalmazoknak nevezik  $F = \{F_1, \dots, F_k\}$ . Azt, hogy egy termék vagy termékhalmaz gyakori-e, egy előre megadott érték, a gyakoriságküszöb ( $\sigma$ ) segít eldönteni. Általában százalékos formában adják meg, és azt jelöli, hogy az összes tranzakció hány százalékában szükséges, hogy szerepeljen egy elem vagy elempár ahhoz, hogy gyakori legyen.

A fenti módszer nagyon sokféle kérdésre válasszal, problémára megoldással szolgálhat, mint például a kockázatelemzés, vagy bármely olyan esemény, ahol a tranzakciónak és elemeknek megfeleltethetőek az előforduló tényezők vagy esetek. A kapott asszociációs szabályok pedig egy vállalat stratégiájában hatékonyan felhasználhatók akár profitnövelés céljából is. Tudományos munkánk alapgondolatát a gazdaságinformatika terén is elterjedt vásárlói kosár elemzés teljesen újszerű megközelítése adta, beleértve a kosarak vizuális megjelenítését, és az algoritmusok továbbgondolását. Mivel elsők között a vásárlói kosarakra próbáltak ilyen irányú megfigyeléseket, algoritmusokat alkalmazni, így az ott használt fogalmak maradtak a problémakör alapfogalmai (tranzakció, elem, stb.). Törekszünk arra, hogy ezeket a fogalmakat a megfelelő kontextusban használjuk, és bár a hétköznapi életben más

jelentésekkel is rendelkeznek, a gyakori elemhalmaz keresésnél mégis egyértelműen és egységesen jelölik a felhasználandó, vizsgálandó adatokat.

Az algoritmusok számtalan gyakori elemhalmazt generálhatnak, melyek elemzése és értelmezése nem mindig egyértelmű, és a szabályok feltárása bonyolult, hosszadalmas, sok munkával járó folyamat. Elmondható tehát, hogy a gyakori elemhalmazok értelmezést, és utófeldolgozást különösen igényelnek (Han - Kamber, 2006.), hiszen ezek nélkül értelmetlenné válik feltárásuk. A vizualizációs algoritmusok használata az elemek és tranzakciók közötti összefüggések szemléletesebb, látványosabb áttekintésében segítenek, megkönnyítve a megértésüket, és hamarabbi felfogásukat. Dolgozatunk célja olyan intuitív eszköz létrehozása, mely könnyen kezelhető, az adathalmazból gyorsan és hatékonyan nyeri ki a gyakori elemhalmazokat, és amely a kapott eredményeket és tranzakciókat szemléletesen jeleníteni meg.

Az adatok kinyerésén túl számos lehetőséget találunk a megfelelő megjelenítésére is. Az egyszerűbbektől, mint a doboz diagram vagy a dendrogram, egészen széles a választék a bonyolultabb megjelenítési módszerekig. Ilyen a Sammon leképezés (Mátyus, 2005.), a polimerizáció (Pach, 2007.) vagy Tanimoto hasonlóságmértéken alapuló vizualizáció (Boztuğa - Reuttererb, 2008). Li Yang és D. Bruzzese a párhuzamos koordinátákat használták fel a gyakori elemhalmazok és asszociációs szabályok megjelenítéséhez, továbbá Bruzzese többszörös összefüggési analízis segítségével 1-halmazokat jelenített meg (Yang, 2003./ Davino – Bruzzese, 2003.). Vannak olyan módszerek is, melyek az adatokat önmagukban próbálják megjeleníteni, ilyen a pontdiagram mátrix, párhuzamos koordináták módszere, cherno ábra, pálcikás megjelenítés (Kriegel - Keim, 1996.).

A megfelelő csoportosításhoz klaszterezzük a már előzetesen kinyert gyakori elemhalmazokat (bicluster segítségével), majd a vizualizációhoz az úgynevezett VAT (Visual Assessment of (Cluster) Tendency) algoritmust használjuk fel, amit ezelőtt még senki nem alkalmazott hasonló problémára. A klasszikus csoportosítási eszközökkel ellentétben nem külön - külön, hanem szimultán vizsgáljuk az eseményeket, és szimultán tárjuk fel az egymáshoz hasonló elemhalmazokat és az egymáshoz hasonló tranzakciókat.

Egy már meglévő bittable technika (Abonyi, 2006.) újragondolásával egy újfajta megközelítést kapjuk a gyakori elemhalmazok kinyerésének, felhasználva egy olyan mátrixműveletet, mely segítségével számos elempárról már az elemzés elején megállapíthatjuk, hogy bizonyosan nem lesznek elemei a gyakori elemhalmazoknak.

Ezzel a lépéssel a vizsgálandó halmazt egy kisebb halmazra tudjuk redukálni, ezáltal megspórolva olyan felesleges iterációkat, melyek analizálásával biztosan nem kapnánk hasznos elem párokat. A vizsgálandó halmaz méretbeli csökkenésével természetesen az algoritmus futási ideje is csökken, így hatékonyabban és gyorsabban kapunk eredményt. A technika továbbá felső becslést is ad a fellelhető legnagyobb gyakori elemhalmaz nagyságára.

Dolgozatunkban feltárjuk a kevésbé használt Bittable algoritmus alkalmazási lehetőségeit, és erre egy egyszerűen interpretálható és implementálható programot készítünk MATLAB-ban, amely ötvözi a VAT, a Biclustering és a Bittable algoritmusok sajátosságait.

Tudományos munkánk a bioinformatikai biclustering, és a co-clustering technikák felé is mutat, hiszen ezek az algoritmusok szintén a gyakori elemhalmaz keresésre, és az általunk használt algoritmusokra vezethetők vissza, megfeleltetve az elemeknek a gének, szövetek tulajdonságait, tranzakcióknak pedig az egyes szöveteket, mintákat. További lehetőséget nyújthat a mérnöki alkalmazásoknál is, mint például az incidens-menedzsmentnél (alarm management) a probléma felügyeleti stádiumban, ahol a bekövetkezett hibák okát kutatják, továbbá akár riasztást, figyelmeztetést is adhat, ha olyan események együttállását érzékeli, melyek nagy valószínűséggel meghibásodáshoz, incidenshez vezethetnek.

Munkánk végső eredménye egy Matlab alkalmazás, melynek inputja a különböző tranzakciók, outputja pedig az ezekből nyert gyakori elemhalmazok, és ezek vizuális megjelenítése. A programot különböző méretű bemenetekre, azaz különböző méretű tranzakciókra lehet alkalmazni, függetlenül attól, hogy ezek vásárlói kosarak-e, vagy pedig egyéb tranzakcióknak megfeleltethető adatok. Így a programot elég széles körben lehet alkalmazni, és számos problémára megoldást nyújthat.

Dolgozatunk 2. fejezetében rövid áttekintést nyújtunk az adatbányászatban ismert vásárlói kosarakról, azok elemzési lehetőségeiről, valamint célkitűzéseinket ismertetjük. A 3. fejezetben a bittable technikát részletezzük alaposabban, valamint kitérünk az Apriori-elvre is (Megj.: A kifejezés az „a priori” latin szóból ered, jelentése: „abból, ami korábban van”. Írásmódja a tudományos cikkek között is különböző. Használják külön „a priori” –ként, egyben „apriori”-ként, valamint a dolgozatban használt, nagy betűs írással. Utóbbira példa az Abonyi János által szerkesztett Adatbányászat- a hatékonyság eszköze című könyv, mely indító szerepet játszott abban, hogy végül adatbányászati technikáról íródott a dolgozat.). A 4. fejezetben ezen

algorithmus eredményeként megkapott mátrix klaszterezéssel történő rendezéséről, valamint vizualizációjáról teszünk említést. Az 5. Kísérleti rész című alfejezetben egy konkrét algoritmus megvalósítását, és annak illusztrálását fejtjük ki, továbbá említést teszünk mérnöki alkalmazásokban való lehetőségekről. A 6. alfejezetben összegezzük a tapasztalatainkat, a 7. fejezetben pedig a felhasznált irodalmak listája olvasható.

## 2. Gyakori elemhalmazok

A gyakori elemhalmazok feltáráshoz első lépésben szükség van egy adatbázisa vagy adatsorokra, melyek különböző elemeket tartalmaznak, és amelyből a gyakori elemhalmazok kinyerhetők (Abonyi, 2006). Jelölje  $P = \{P_1, \dots, P_n\}$  az adatbázisban előforduló elemek halmazát. Az  $E$  (általános, különböző elemeket tartalmazó) halmaz részhalmazait elemhalmazoknak nevezzük. Ha egy elemhalmaz  $k$  darab elemet tartalmaz, akkor  $k$ -elemhalmaznak hívjuk (Balambér, 2011). Legyen  $T = \{T_1, \dots, T_N\}$  tranzakciók (rekordok) halmaza, ahol minden  $T_i \in T$ ,  $i \leq n$ . A  $T$  tranzakciók azon számát, amelyekben az  $X$  elemhalmaz előfordul, az  $X$  elemhalmaz gyakoriságának nevezzük. Számolására az alábbi képletet használjuk:  $gyakoriság(X) = |\{T_i \mid X \in T_i, T_i \in T\}|$ . (A gyakorlatban sokszor százalékos formában adják meg a gyakoriságot, azaz az előfordulás és a tranzakciók számának hányadosaként.) A gyakoriság tehát nem más, mint a  $P(X)$  valószínűség.

| Tranzakciók | Elemek                    |
|-------------|---------------------------|
| 1           | tej, tojás, kenyér        |
| 2           | kenyér, pelenka, sör      |
| 3           | tej, pelenka, kenyér, sör |
| 4           | pelenka, sör              |

2.1. táblázat: Vásárlói kosarak

Az  $X$  elemhalmazt gyakorinak hívjuk, ha a tranzakciók felhasználó által megadott számában (százalékban) előfordul (Bodon, 2010./ Hand – Manilla - Smyth, 2001). Ezt az úgynevezett minimális gyakoriságot vagy gyakoriságküszöböt  $\sigma$ -val jelöljük. Azaz  $X$  gyakori, ha  $gyakoriság(X) \geq \sigma$ . Szemléltetésként tekintsük a 2.1-es táblázatot, amelyen különböző vásárlói kosarakat láthatók. Egy vásárlói kosár felel meg egy tranzakciónak. A táblázatból kiolvasható, hogy  $E = \{tej, kenyér, pelenka, tojás, sör\}$  az elemek, termékek halmaza (Anbarasi – Srinivas - Vivekanandan. 2011). A kételemű,  $\{pelenka, sör\}$  elemhalmaz gyakorisága 3, azaz 75%, mert a második, harmadik és a negyedik tranzakció is tartalmazza mind a két terméket. A gyakori elemhalmazok hiányossága, hogy az elemhalmazokban

szereplő elemek közötti kapcsolatot nem mutatják. A kapcsolatok kifejezésére külön lehet asszociációs szabályokat generálni, de mivel dolgozatunk fő témáját nem ez adja, így a részletezésében való elmerülést mellőzük.

A fenti példánál a gyakori elemhalmazok megtalálása egyszerű, hiszen könnyedén végigszámolhatjuk az összes elemre, hogy megfelelnek-e a küszöbszintnek, és gyakoriak-e. A párok megtalálása sem jelent különösebb gondot, mivel kevés adattal kell dolgoznunk. Nagyobb adatbázisoknál viszont rengeteg időbe telne, míg az összes variációt előállítanánk, és egyesével vizsgálnánk gyakoriság szempontjából. Ezt küszöböli ki az az anti-monoton tulajdonság, ami a gyakori elemhalmaz kereső algoritmusoknál megtalálható: Kimondja, hogy ha egy elemhalmaz nem gyakori, akkor az, egy újabb elemmel bővítve sem lesz gyakori. Ezt az állítást megfordítva kapjuk, az Apriori-elvet, azaz, ha egy elemhalmaz gyakori, akkor a rész-elemhalmazai is gyakoriak lesznek. Ezt az elvet használjuk fel dolgozatunkban, és az elemhalmaz kereső algoritmusunkban is központi szerepet kap.

Az Apriori-elvhez kapcsolódó fogalom a zárttság fogalma, amely szerint az  $X$  elemhalmaz zárt, amennyiben nincs nála bővebb halmaz, amelynek támogatottsága megegyezik  $X$  támogatottságával. Tehát, ha egy új elemet adunk a gyakori elemhalmazunkhoz, a kapott elemhalmaz gyakorisága minden esetben különbözni fog tőle.

Mivel a gyakori elemhalmaz keresési probléma eléggé általános ahhoz, hogy széles skálán alkalmazható legyen más területeken is, ezért a módszere és fejlesztésére irányuló szakirodalmak tárháza igen nagy. Legtöbbször az Apriori-elvvel és algoritmussal kapcsolják össze, és szinte kivétel nélkül tartalmazznak rekurziót (Hand – Manilla – Smyth, 2001.). Jó részük a számításokat általános vagy bináris mátrixokkal végzik, ezáltal is meggyorsítva a végrehajtás folyamatát, és kiaknázva a mátrixműveletek lehetőségeit (Abonyi, 2006./ Tan – Steinbach – Kumar, 2006./ Ren – Wang – Wang – Feng 2011.).

### 3. A Bittable algoritmus

A gyakori elempárok kinyerésére számos lehetőséget kínál a szakirodalom. Ezek közül az egyik leggyorsabb a Bittable algoritmus. Képezzük  $\mathbf{B}$  bináris mátrixot a  $T$  (tranzakciók) és  $P$  (termékek) segítségével úgy, hogy a sorokban (melyeknek száma  $N$ ) a tranzakciók, az oszlopokban (melyeknek száma  $n$ ) a termékek szerepeljenek. Ha  $T_i$  tartalmazta  $P_j$  terméket, akkor  $\mathbf{B}_{ij} = '1'$ , különben  $\mathbf{B}_{ij} = '0'$ , így megkapjuk  $B_{Nn}$ -t, melynek a mérete  $N \cdot n$  lesz. A legenerálás végén egy bináris mátrixot kapunk, melynek sorai a tranzakciók, oszlopai az elemek lesznek.

$$B = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

A 2.1-es táblázatnak megfelelő bináris mátrix

(3.1.)

Vegyük  $\mathbf{B}'$  mátrixot, melyet a  $\mathbf{B}$  transzponálásával kapunk.

$$B' \cdot B = \begin{bmatrix} 2 & 2 & 1 & 1 & 1 \\ 2 & 3 & 1 & 2 & 2 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 2 & 0 & 3 & 3 \\ 1 & 2 & 0 & 3 & 3 \end{bmatrix}$$

2.1-es táblázat Bittable mátrixa

(3.2.)

A Bittable algoritmus az újonnan kapott  $(\mathbf{B}' \cdot \mathbf{B})$  mátrix elemzésén alapszik. Itt a főátlóból leolvasható, hogy a  $P_{ij}$  elem hányszor fordult elő összesen a tranzakciókban, a főátló alatti (vagy feletti) részből pedig, hogy az adott elempárok hányszor szerepeltek együtt egy tranzakcióban. Amennyiben ez az érték eléri a gyakoriságküszöböt, úgy  $i$  és  $j$  elem együtt gyakoriak lesznek, és gyakori elemhalmazt képeznek. A további gyakori elemhalmazok megtalálásához venni kell  $(i \cdot j)$ -t, és ezeket hasonlítani a többi adatsorral. Az algoritmus végén megkapjuk az összes gyakori elemhalmazt.

### 3.1. Mátrixok szorzásán alapuló hatékonyságnövelő technika

A bittable algoritmus hatékonysága és gyorsasága annak köszönhető, hogy a kettes (majd több) gyakori elempárokat egy lépésben, azonnal megtalálja; értelmezése és programmal történő megvalósítása pedig viszonylag egyszerű és könnyű.

A bináris mátrix és annak transzponáltjából nem csak a fent bemutatott Bittable algoritmus segítségével lehet a gyakori elemhalmazokat kinyerni az adatsorokból.

Rájöttünk, hogy a fordított szorzási sorrenddel  $(\mathbf{B} \cdot \mathbf{B}')$  kapott mátrixot is hasznosan fel lehet használni elemzés szempontjából. Képezzük új, szimmetrikus mátrixot (mérete  $N \cdot N$ ), az eredeti bináris mátrix és annak transzponáltjának szorzatával:  $(\mathbf{B} \cdot \mathbf{B}')$ , ahol a  $(\mathbf{B} \cdot \mathbf{B}')$ <sub>ij</sub> érték megadja  $T_i$  és  $T_j$  tranzakciók metszetének elemszámát, azaz  $(\mathbf{B} \cdot \mathbf{B}')$ <sub>ij</sub> =  $|T_i \cap T_j|$ . Ez a mátrix a nagy mérete miatt nem kerül realizálásra, mivel túl időigényes, és túl sok helyet foglalna el a memóriában.  $(\mathbf{B} \cdot \mathbf{B}')$ <sub>ij</sub> =  $|T_i|$ , ahol  $i = j$ . Tehát az átlóban szereplő értékek megadják az egyes tranzakciók számosságát, elemszámát.

$$(\mathbf{B} \cdot \mathbf{B}') = \begin{bmatrix} 3 & 1 & 2 & 0 \\ 1 & 3 & 3 & 2 \\ 2 & 3 & 4 & 2 \\ 0 & 2 & 2 & 2 \end{bmatrix}$$

A 2.1-es táblázatra alkalmazott újfajta Bittable algoritmus (3.3.)

Mivel gyakorlatilag ez a különböző tranzakciósorokat hasonlítja egymáshoz, a főátlóból kiolvasható, hogy adott tranzakciók hány különböző elemet tartalmaztak összesen. A sorok ( $i$ ) és oszlopok ( $j$ ) metszetéből pedig azt, hogy két vásárlónak, vagy két tranzakciónak hány közös eleme van. Azaz  $(i, j)$  jelöli, mennyi a kettő metszetének számossága. A főátló alatti (vagy feletti) részből megsejthető – a legnagyobb előforduló számot megkeresve –, hogy a legnagyobb elemszámú gyakori elemhalmaz mérete maximum mennyi lehet.

Legyen  $(\mathbf{B} \cdot \mathbf{B}')_{ij} = 0$ , ahol  $i \leq j$ , így csak a mátrix alsó háromszögében maradnak érvényes értékek. Legyen  $bmax$  a  $(\mathbf{B} \cdot \mathbf{B}')$  mátrixban szereplő legnagyobb érték. Gyűjtsük ki csoportokba ( $IT\{k\}$ ) a tranzakciókat párosával úgy, hogy léptessünk egyesével egy általános változót ( $k$ ) 1-től  $bmax$ -ig, és vizsgáljuk közben  $(\mathbf{B} \cdot \mathbf{B}')$

mátrixot; ahol  $\mathbf{B}_{ij} \geq k$  gyűjtsük ki, és vizsgáljuk meg, hány különböző tranzakciót érint. Ezeket együtt tegyük  $IT\{k\}$  halmazba.

### 3.2. Mátrixok szorzásán alapuló hatékonyságnövelő technika elemzése

Tegyük  $B$  mátrix segítségével  $Bitems\{\}$  halmaz első elemébe azon termékeket, amelyeket többször vagy ugyanannyiszor vettek, mint a támogatottsági szint, azaz  $\sigma$ . Tehát  $Bitems\{1\} = \{\text{ahol } szum(B) \geq \sigma\}$ . Legyen  $Bi\{1\}$  a  $Bitems\{1\}$ -ben lévő termékek pontos támogatottsága tranzakciónként. ( $Bi\{1\} = Bitems\{1\}$ ) Ezután egy általános változót ( $k$ ) kezdünk léptetni 2-től  $bmax$ -ig, és minden léptetéskor a következőket hajtjuk végre: egy index változóba kigyűjtjük az alábbiakat:  $\{0, X, Y\}$ , ezek jelölik a számunkra hasznos elemek indexét. A 0 indexű elem mindenképpen benne lesz, az a kiinduló elem.  $Y$ : az előző  $Bitems\{k-1\}$  halmaz mérete, azaz az utolsó elem indexe.  $X$ : a számunkra hasznos egyedi indexek.

A meghatározott indexeket párosával kezdjük vizsgálni.  $v$ -be kigyűjtjük őket, hogy ne kelljen túl nagy mátrixot szorozni, majd  $m$  változóba tároljuk a  $Bi\{k-1\}$ -ből  $v$  segítségével kigyűjtött elemek és ezek transzponáltjainak szorzatát. Mivel szimmetrikus mátrixról van szó, elegendő csak az alsó vagy felső háromszöget vizsgálni. Megkeressük,  $m$ -bel hol vannak olyan elemek, melyek megfelelnek a támogatottsági szintnek.

A  $Bitems\{k\}$  halmazt folyton növeljük a  $Bitems\{k-1\}$ -ből  $v$  és  $[dum1, dum2]$  segítségével vett elemekkel.  $Bi\{k\}$ -t szintén növeljük a  $v$  és a  $[dum1, dum2]$  segítségével. Végül  $Bi\{k\}$ -t rendezzük növekvő sorrendben, ezek után növeljük  $k$  értékét. Az algoritmus pszeudokódja az 8.1. számú mellékletben látható.

Másmilyen megközelítésből: legyen  $max$  a főátló alatti részben előforduló legnagyobb szám; a főátlót és az a feletti részt a továbbiakban nem tekintjük. Tegyük  $D$  halmazba azokat az indexpárokat, ahol  $B_b(i, j) = (max)$ . Amennyiben  $D$ -ben a gyakoriságküszöböt nem elérő számú különböző indexek találhatóak, biztosak lehetünk benne, hogy a legnagyobb elemszámú gyakori elemhalmaz mérete kisebb, mint  $max$ . Ellenben, ha ezek elérik a gyakoriságküszöböt, elegendő csak az indexeknek megfelelő tranzakciókat összehasonlítani. Ha itt legalább a gyakoriságküszöbször egyezik meg

max termék, akkor azok a termékek képezik a legnagyobb elemszámú gyakori elemhalmazt. Az Apriori-elvet felhasználva ezek részalmazait is gyakori elemhalmazoknak tekinthetjük.

Az algoritmus folytatásában a max-tól a gyakoriságküszöbig vizsgáljuk az elemhalmazokat, így a végén ez által is megkapjuk az összes gyakori elemhalmazt. Az új algoritmus hatékonysága a kizáráson alapszik, azaz az Apriori-elvből kiindulva kizárjuk azon elemek vizsgálatát, amivel biztosan nem kapunk gyakori elemhalmazokat, valamint egy nagyobb gyakori elemhalmaz részalmazait automatikusan gyakoriaknak tekintjük. A legtöbb algoritmussal ellentétben itt nem a kettes-párok vizsgálatától indulunk, és haladunk a nagyobb elemszámúakig; hanem a legnagyobb elemszámúval indulunk, és haladunk a kisebb elemszámúak felé.

A Bittable algoritmus megismeréséhez elengedhetetlen az Apriori algoritmus ismerete, hiszen ez adja a Bittable algoritmus alapköveit. Az Apriori algoritmus pszeudokódja az alábbi:

```

k = 1
(Ck;DF) = Transform(D)
Fk = Count(Ck, DF,  $\sigma$ )
while |Ck|  $\neq$  0 do
  inc(k)
  Ck = Generate(Fk-1)
  Ck = Prune(Ck)
  Fk = Count(Ck, DF,  $\sigma$ )
  F = F  $\cup$  Fk
end

```

3.4. ábra : Gyakori elemhalmazok kinyerése  
( $\sigma$  = minimális támogatottság, D = adatbázis)

### 3.3. Az Apriori algoritmus

Az algoritmus a nevet az Apriori-elvről kapta, amely kimondja, hogy a  $(k+1)$ -elemű jelölteket a  $k$ -elemű elemhalmazból állítjuk elő, azaz egy korábbi ismeretet (a  $k$ -elemű elemhalmazokat) használ fel az újabb szinten lévő jelöltek (a  $(k+1)$ -elemű jelöltek) előállítására.

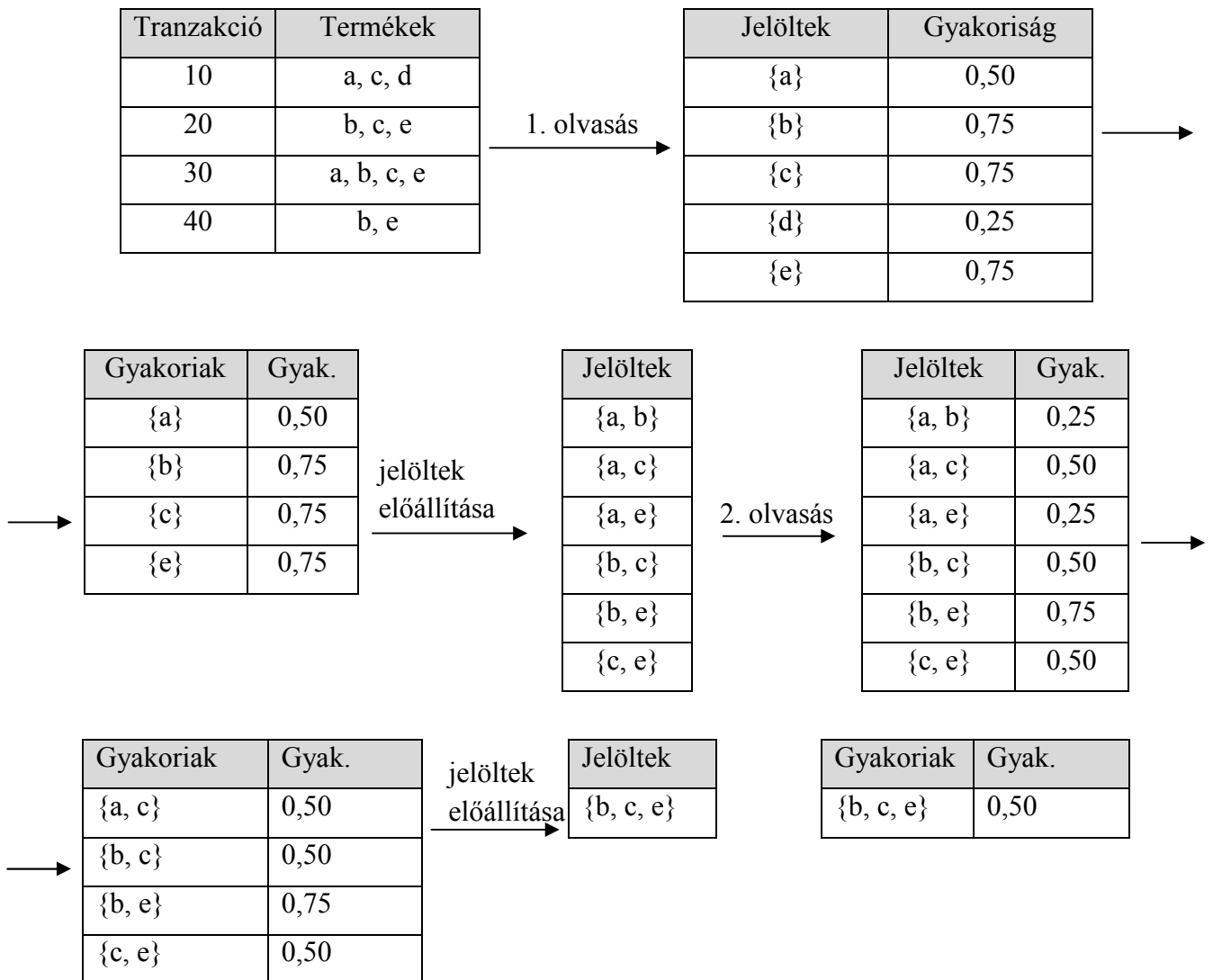
Az algoritmus iteratív, mely szintenként határozza meg a gyakori elemhalmazokat. Minden egyes szinten két fázis van: jelölt generálás és gyakoriság ellenőrzés. Az első fázisban az előző szinten talált gyakori  $(k-1)$ -elemhalmazokból állítjuk elő a  $k$ -elemű jelölteket. Az új jelöltek előállítását a következő egyszerű

illesztési művelettel végezzük: két gyakori  $(k-1)$ -elemhalmaz összeilleszthető, ha a lexikografikusan rendezett első  $k-2$  elemük megegyezik, és az utolsó elemük különböző.

Példaként tekintsük a következőt: legyen két gyakori elemhalmazunk, a {sör, pelenka, tej}, valamint a {sör, pelenka, kenyér}. A két elemhalmaz összeilleszthető, mivel az első két elemük megegyezik, és az utolsó eltérő. Az új jelölt tehát a {sör, pelenka, tej, kenyér} lesz.

A másik fázisban beolvassuk az adatbázist tranzakcióról tranzakcióra, és növeljük azon jelöltek gyakoriságát, melyek az adott tranzakcióban szerepelnek. Az algoritmus működése akkor ér véget, ha nem tud új jelöltet előállítani, vagy az előállított jelöltek között egyik sem éri el a felhasználó által megállapított gyakoriságküszöböt.

Az algoritmus működésére tekintsük az alábbi példát:



3.1. táblázat: Apriori algoritmus szemléltetése

## 4. Bittable vizualizációja

### 4.1 Biclustering

A biclustering, ahogy neve is sejteti (klaszterezés = csoportosítás), egy fajta csoportosítás, melynek lényege, hogy egy mátrix sorainak és oszlopainak sorrendjét felcserélve olyan összefüggő struktúrát, ún. biclustert hozunk létre, mely alkalmas arra, hogy ennek segítségével feltárjuk az elemek közötti rejtett összefüggéseket (Gu – Liu, 2008. / Mitra – Banka, 2006. / Zhang – Teo – Ooi – Tan, 2004). Ez a technika alkalmazható a gyakori elemhalmaz keresésnél, a már korábban említett  $B$  mátrix sorainak és oszlopainak megfelelő cseréjére, mely csere eredményeképp az együtt gyakori termékek, termékhalmozok a mátrixon belül egy helyen, területileg egy kupacban helyezkedjenek el. A sor- és oszlopcsere következtében a mátrix tartalmi jelentése nem változik, azaz a tranzakciók továbbra is ugyan azokat az elemeket fogják tartalmazni, így továbbá is relevánsak maradnak, csak a felírásuk történik másmilyen sorrendben.

Biclustering definíció: A  $(G, C) \in 2^{\{1, \dots, n\}} \times 2^{\{1, \dots, N\}}$  párt *maximális befogadó képességű* biclusternak nevezzük akkor, és csak akkor, ha (1)  $\forall i \in G, j \in C : e_{ij} = 1$  és (2)  $\nexists (G', C') \in 2^{\{1, \dots, n\}} \times 2^{\{1, \dots, N\}}, (i) \forall i' \in G', j' \in C' : e_{i'j'} = 1$  és (ii)  $G \subseteq G' \wedge C \subseteq C' \wedge (G', C') \neq (G, C)$

Vegyük példának a következő mátrixot:

$$B = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

(4.1.)

Az átalakítást követően a következő mátrixot kapjuk:

$$B = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \end{pmatrix} \quad (4.2.)$$

A biclustering eljárás során tehát addig cseréljük az oszlopokat és sorokat, míg olyan részmátrixokat nem kapunk, melyek vagy csak 1-es, vagy csak 0-s számjegyeket nem tartalmaznak. A fenti mátrix jobb felső részmátrixa csak 0-s számjegyet tartalmaz, mely azt reprezentálja, hogy azokhoz az elemekhez nem tartozik tranzakció. Az algoritmus futása során ezen elemek összehasonlítását mellőzhetjük, így jóval hamarabb lefut az iteráció. Az algoritmus a  $B$  mátrix többi részén végighalad rekurzívan mindaddig, míg például csak 1-esekből álló biclustert nem reprezentál. Ha létrejön kettő olyan bicluster, mely nem tartalmaz közös sort, vagy oszlopot, akkor a két részmátrixot egymástól függetlenül feldolgozhatjuk.

Itt fontos megjegyezni, bár az Apriori-elv kimondja: minden gyakori elemhalmaz részhalmazai is gyakoriak lesznek, a klaszterezés során mégsem feltétlenül a nagyobb gyakori elemhalmazzal dolgozunk. Hiszen lehet, hogy egy részhalmazának a támogatottsága sokkal nagyobb lesz, mint magáé a halmazé, és méretétől függően sokkal nagyobb jelentőséggel bírhat a kisebb elemhalmaz, ha előfordulása jelentősen nagyobb.

## 4.2 Az adatok megjelenítésének fontossága

A klaszterezés, vagyis az oszlopok és sorok megfelelő irányú cseréje a megjelenítésnél játszik fontos szerepet. A már előre csoportokba rendezett halmazok vizuális kivetülése sokkal látványosabban és szemléletesebben jellemzi az elemeket, és azok egymáshoz való viszonyát. Míg ezzel szemben egy rendezetlen halmaz vizualizációjából még a gyakorlott mérnöki szem sem tudná kiszűrni, vagy csak nehezen, mely termékek melyekkel lehetnek párban gyakoriak. Tehát kimondhatjuk, hogy a megfelelő megjelenítés legalább annyira fontos, mint maga a gyakori elemhalmazok kinyerése, hiszen egy megfelelő ábráról szinte azonnal észrevehető számos lényeges adat, ezáltal segítve a pontos és gyors asszociációs szabályok felállítását.

Az elemzés szempontjából fontos még az elemhalmazok hasonlósága is, azaz, hogy a különböző elemek hogyan, miként függnek egymástól, mennyire helyettesítő vagy kiegészítő termékekről van szó. Kiegészítő termékeket (ilyen pl. a kávé és a sűrített tej) gyakrabban vásárolnak együtt, így a gyakori elemhalmaz keresésnél többször kapunk ilyen termékeket eredményül. Az asszociációs szabályok feltárásakor ezeknél a termékeknél könnyű dolgunk van, hiszen sokszor az egyik terméket nem is fogyasztják a másik nélkül. A kiegészítő termékeken túlmenően a megfelelő megjelenítésből kivehető az is, hogy a többi termék hogyan viszonyul egymáshoz, milyen mértékben vásárolják őket együtt. Ez hasznos lehet különböző marketingstratégiáknál, vagyis, ha  $a$  termék vásárlása esetén  $b$  terméket is előszeretettel választják a vásárlók, akkor  $b$  termék  $a$  mellé helyezésével növelhető előbbi eladása. Egy másik stratégia lehet a két termék egymástól távol való elhelyezésével más termékek vásárlásának ösztönzése.

A fenti marketinges szemléltetéseken kívül a megfelelő vizualizáció a mérnöki alkalmazásoknál is legalább ugyanolyan fontos. Amennyiben a tranzakciók elemei különböző reakciók, vagy különböző események bekövetkezése, a pontos megjelenítéssel akár súlyos eseteket is meg lehet előzni, hiszen az eredményül kapott ábrából könnyen leszűrhető, hogy a kritikus események mely másik eseménnyel, és mikor fognak együtt fennállni.

A tranzakciók hasonlósága szintén fontos, hiszen ezt használjuk fel a Bittable algoritmus átértelmezésében. Megjelenik, hogy a különböző tranzakciók vagy különböző vásárlók hány közös terméket vettek, mennyi a közös halmazuk elemszáma,

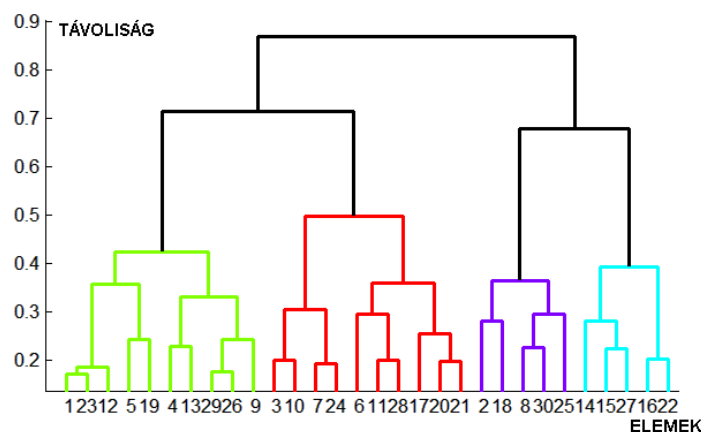
illetve, hogy melyek ezek a termékek, elemek. Ez, ahogy az algoritmusnál már említettük, előrejelzéseket adhat a gyakori elemhalmazok számosságára, és további számos dolgot vetíthet előre, mint például a legnagyobb számosságú gyakori elemhalmaz mérete, és a klaszterezésnek köszönhetően ugyan úgy megjelenik a vizualizációnál, mint ahogy az elemek hasonlósága.

Elmondható tehát, hogy a megfelelő vizualizációval mind az elemek, mind pedig a tranzakciók hasonlósága ideálisan szemléltethető, kiemelve és erőteljesebben láttatva a fontosabb eseteket.

### 4.3 Az adatok megjelenítése VAT algoritmussal

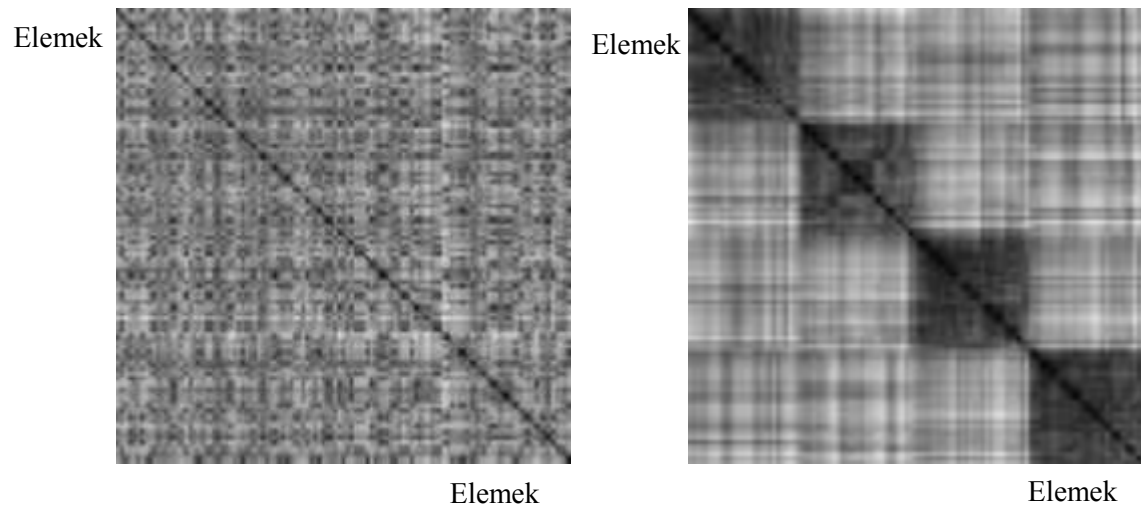
A gyakori elemhalmazok vizualizációjára számos megoldást kínál a szakirodalom, mi ezek közül a Visual Assessment of (Cluster) Tendency (továbbiakban VAT) algoritmust választottuk (Bezdek - Hathaway, 2002). Ahogyan neve is sugallja, ez az eljárás megfelelően elkülönítve jeleníti meg a hasonló elemhalmazokat és tranzakciókat, azaz a klaszterek tendenciáját.

Az algoritmus azt méri, hogy a különböző tranzakciók mennyire egyformák, vagy különböznek egymástól. A Tanimoto hasonlóságot felhasználva egy hasonlósági mátrixot generál, értékeit pedig 0 és 1 közötti intervallumba eső számmal tölti fel ( $S_{ij} = [0,1]$ ), ahol 0 a teljesen különböző 1 pedig a teljes mértékben hasonló tranzakciókat jelöli. A dendrogramokhoz hasonlóan először a legkisebb távolságú elemeket találja meg, majd sorban az egyre távolabbiakat, végül az így kapott sorrendet a szimmetrikus mátrix átlójában ábrázolja. Az egymáshoz jobban hasonlító tranzakciók megjelenítése sötétebb, míg az egymáshoz kevésbé hasonlóak színe világosabb pixelegységekkel történik, így jobban érzékeltetve a hasonlóságokat.



4.1. ábra: Példa adatok dendrogramon való ábrázolására

A VAT eredményeképp tehát egy, az átlójánál különböző mélységben színezett négyzetet kapunk. Ez a négyzet nem más, mint a biclusterezés eredményeként, azaz a megfelelő oszlop- és sorcserékkel kapott mátrixunk, a különböző tónussal színezett pixelblokkok pedig a különböző klasztereket reprezentálják.



4.2. ábra: Egy random és egy rendezett adatsor VAT ábrája

A fenti illusztratív példa nem csak a VAT algoritmus kimenetelét szemlélteti megfelelően, de azt is jól érzékelteti, mennyire fontos az adatsorok megfelelő rendezése. Míg az első ábra nagyjából homogén szürke felületet mutat, a másodikon tökéletesen ki lehet venni a különböző adatcsoportosulásokat, klasztereket, melyekre ezek után gyorsan, hatékonyan lehet a megfelelő asszociációs szabályokat megtalálni.

## 5. Kísérleti rész

Dolgozatunk célkitűzése egy olyan program megalkotására volt, amely bármilyen adatbázisból, ami megfelel a bemeneti struktúrának, leképezi annak a bináris mátrix megfelelőjét, és a továbbiakban ezzel a mátrixszal számításokat végez: bekér egy támogatottsági szintet, és annak megfelelően kiértékeli az adatsorokat a korábban említett bittable technikák segítségével. Az eredményekre zárt elemhalmazokat képez, és a megfelelő klaszterezés és hasonlóságmérték alkalmazásával a VAT algoritmus segítségével ábrázolja őket. Az elkészült program hatékonyságát különböző adatbázisokon való teszteléssel és futtatással ellenőriztük, vizsgálva az idősorokat, valamint összehasonlítottuk az eredményét már meglévő algoritmusokéval. Fontos szempont volt, hogy egy olyan szellemi terméket hozzunk létre, ami egyedi, hatékony, a valós eredményt torzítás és félrevezető információk nélkül ábrázolja, valamint optimálisan felhasználható legyen döntéstámogatás céljából.

### 5.1. A kifejlesztett algoritmusok és program

Dolgozatunk újítása a kifejlesztett algoritmusokat alkalmazó program implementálása. A megvalósításukhoz a célszerűség érdekében MATLAB-ot használtunk, hiszen ez a legalkalmasabb és leghatékonyabb eszköz a különböző mátrixműveletek végrehajtásához.

Első lépésben importál egy tranzakciókat tartalmazó adatbázist, majd bekéri az alkalmazandó támogatottsági szintet.

Az elő-feldolgozás a különböző elemek kigyűjtésével kezdődik. Ezzel automatikusan egy indexszámot kapnak, így a későbbi azonosításuk leegyszerűsödik. Ezt követően deklarálunk egy bináris mátrixot, amit a tranzakciókon való végighaladás közben folyamatosan feltöltünk '1' és '0' értékkel a szerint, hogy az adott tranzakció tartalmazta-e az adott indexű oszlopnak megfeleltethető terméket vagy sem. A ciklus végén így megkapjuk az eredeti tranzakciókkal ekvivalens bináris mátrix formát.

A különböző elemek kiválogatásának forráskódja a 8.2-es mellékletben látható, a bináris mátrixba való konvertálás forráskódja pedig a 8.3-asban.

A bináris mátrix transzponáltjából és az eredeti bináris mátrix szorzásából kapott új mátrixot dolgozzuk fel egy korábbi bittable technikával. A MATLAB lehetőségeit kihasználva nem szükséges külön leképezni a transzponáltat, majd külön szorozni, hanem megengedett, hogy ezt magában a ciklusmagban tüntessük fel műveletként.

A Bittable technika alkalmazásának forráskódja a 8.4-es mellékletben látható.

Az algoritmus lefutás után a gyakori elemhalmazokat adja vissza.

Ez a programrész a kettes gyakori elempároktól indul és folyamatosan találja meg a nagyobb elemszámú gyakori elemhalmazokat. A másik implementált algoritmus, az előbbi továbbfejlesztése is hasonlóképp működik, csak előzetes mátrixműveletek segítségével már az elején kizár olyan lépéseket, melyek biztosan nem adnának gyakori elemhalmazt.

A következő fő lépés a megtalált gyakori elemhalmazok klaszterezése, azaz a sorainak és oszlopainak megfelelő rendezése (A. Serin, M. Vingron. 2011).

```
C=[];
for k=1:length(Ci)
    C=[C Ci{k}];
end
% Similarity - based on your idea / but based on the closed itemset
S1=C*C'; %AND
S2=ones(size(C))*C';
S3=C*ones(size(C))';
S4=max(S2,S3);
S=S1./S4 %Tanimoto similarity

% Similarity of the items
iS1=A'*A; %AND
iS2=ones(size(A))*A;
iS3=A'*ones(size(A));
iS4=max(iS2,iS3);
iS=iS1./iS4 %Tanimoto similarity of the items

iorder=vat_t_vl(1-iS) %This is the VAT of the transactions
```

### 5.1. ábra: A Biclusterezés

Az adatok most már készen állnak a megfelelő megjelenítésre. Erre először a `spy()` beépített függvényt használjuk:

```
spy(bin_matrix(order,iorder))
```

(5.2)

A Tanimoto hasonlóságmérték alkalmazása után a VAT algoritmus használatára külön függvényt definiáltunk, hiszen ez egy eléggé összetett művelet, és a program strukturáltságnak megtartását, és a kódrészlet átláthatóságát így biztosítani tudjuk. Az

algoritmus forráskódja a 8.5-ös mellékletben látható. A program a lefutást követően több ábrán szemléletesen, tagoltan jeleníti meg a gyakori elemhalmazokat.

Úgy véljük, az átdolgozott bittable technika segítségével egy hatékonyabb és gyorsabb algoritmust hoztunk létre a gyakori elemhalmaz keresésre, hiszen a kezdeti kizárások segítségével nagy mértékben csökkenthető a vizsgálandó tranzakciók halmaza, és a méretbeli csökkenés hatására az algoritmus futási ideje is kisebb lesz. A Tanimoto hasonlóságmérték, a klaszterezés és a VAT algoritmus együttes működésével pedig egy megfelelően szemléletes megjelenítést kapjuk az eredményeinknek.

| Feldolgozandó<br>mátrix mérete: | Gyakori elemhalmazok kinyerésének ideje   |                                              |
|---------------------------------|-------------------------------------------|----------------------------------------------|
|                                 | eredeti bittable<br>technikával (seconds) | „javított” bittable<br>technikával (seconds) |
| 15 * 14                         | 0,04                                      | 0,01                                         |
| 100 * 300                       | 0.010157                                  | 0.006447                                     |
| 100 * 147                       | 0.009306                                  | 0.006171                                     |
| 100 * 455                       | 0.012105                                  | 0.008941                                     |

5.1. táblázat: A gyakori elemhalmazok kinyerésére fordított idő szemléltetése

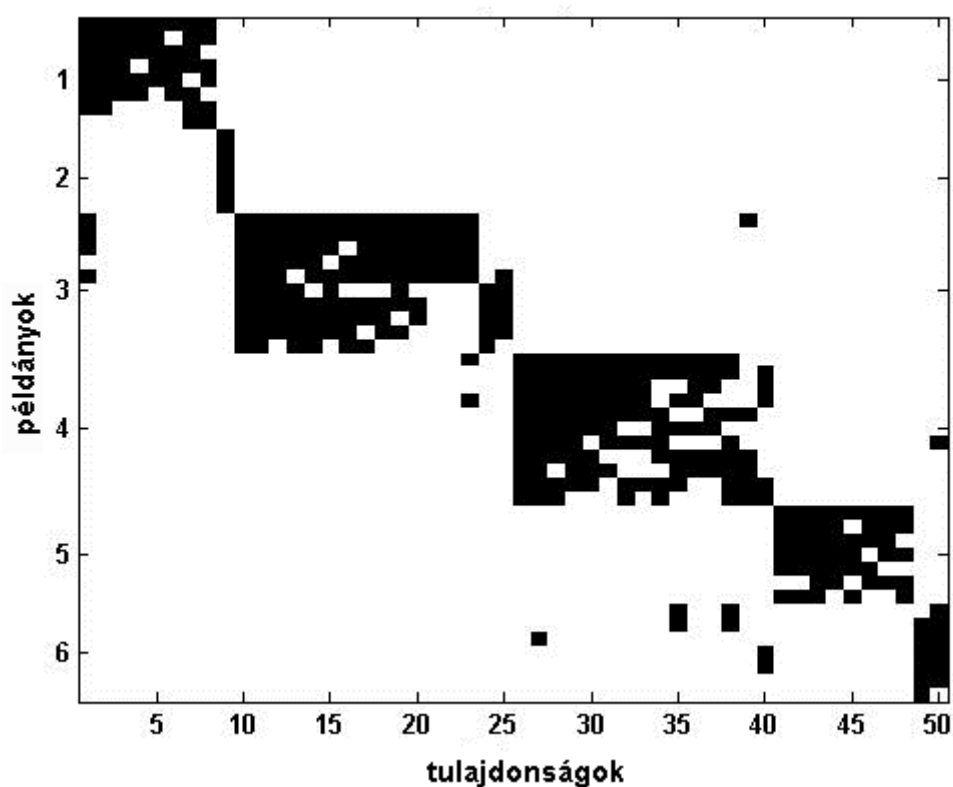
## 5.2. Illusztratív példa

Tekintsünk egy illusztratív példát a program futásáról. A források a <http://adios.tau.ac.il/SpectralCoClustering/> weboldalról származnak.

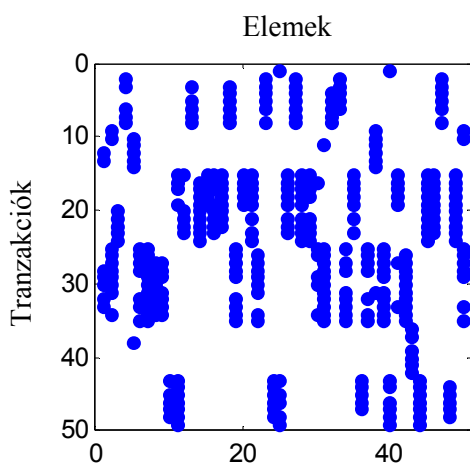
(Dhillon, I.S. (2001) Co-clustering documents and words using bipartite spectral graph partitioning, in proceedings of the acm sigkdd conference, 269-274 )

Az oldal által használt biclusterező algoritmussal kapott ábrát láthatjuk az első képen. A mi algoritmusainkat használva egy sokkal illusztratívabb megjelenítést kapunk, ahol szemléletesebben különülnek el a csoportok, a weboldalon található példához képest.

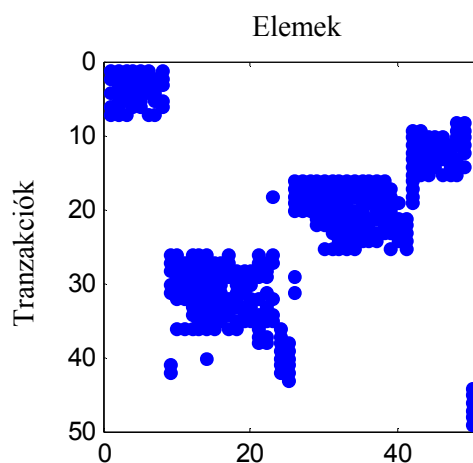
A továbbiakban a programunk által eredményül adott ábrák láthatóak:



5.3. ábra: A weboldal által használt biclusterező algoritmussal generált ábra



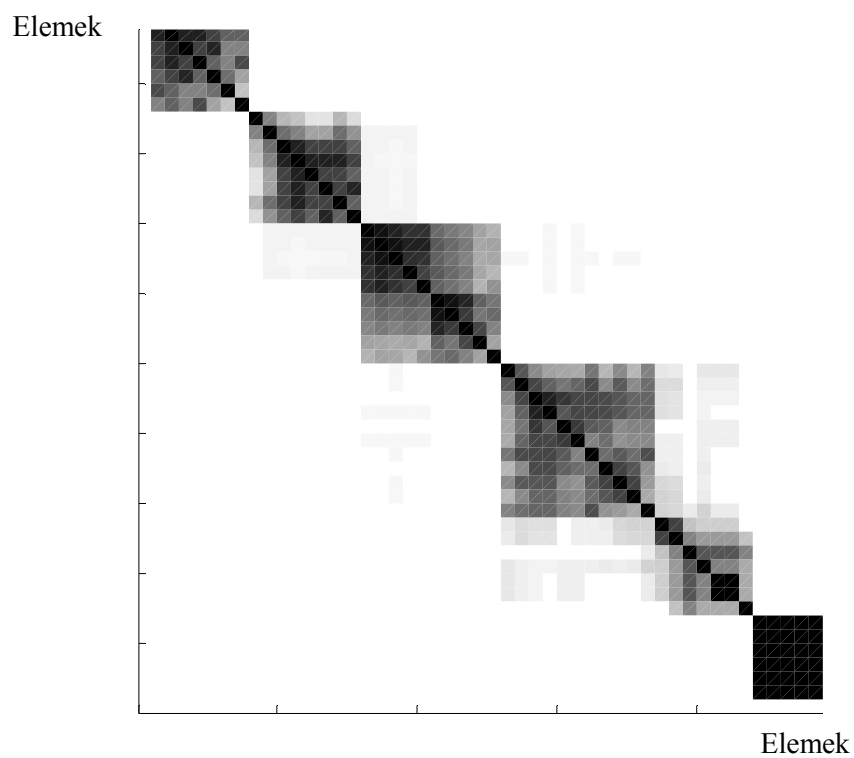
5.4. ábra: Az eredeti tranzakciók



5.5. ábra: Biclusterezéssel való rendezés után

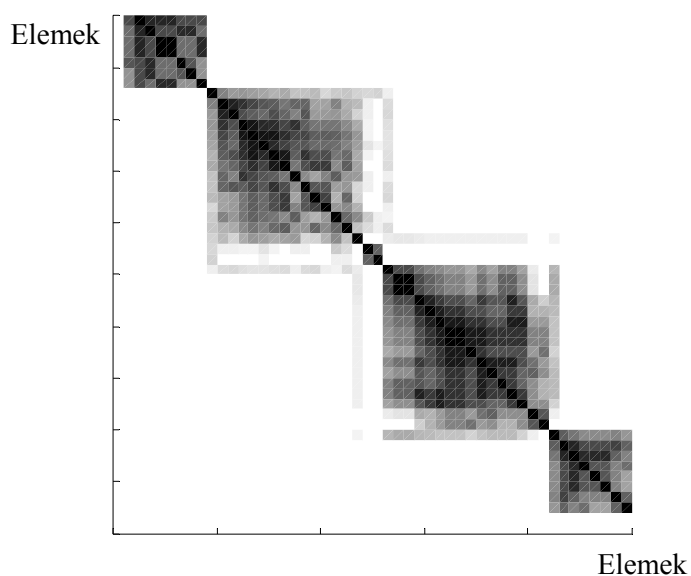
A fenti ábrákon az eredeti tranzakciók halmaza és ezeknek a zárt elemhalmazra való leképezésének megjelenítése látható.

A programunk kimenetelén a tranzakciók VAT algoritmussal generált ábrája jelenik meg, melyen több, jól elkülöníthető csoportot láthatunk.



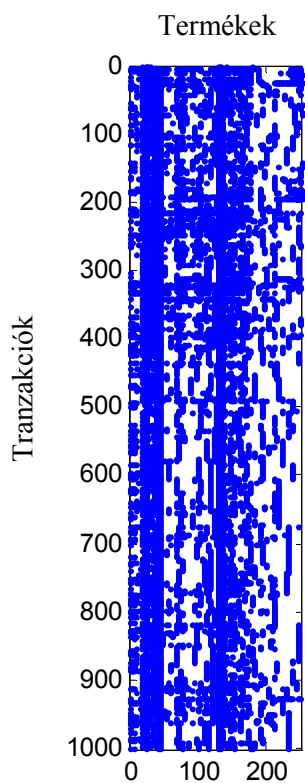
5.6. ábra: A VAT algoritmus kimenetele

Az elemhalmazok VAT-ja, jól mutatja, hogy 4 nagy csoport van, melyek közül két nagyobb rész tűnik ki.

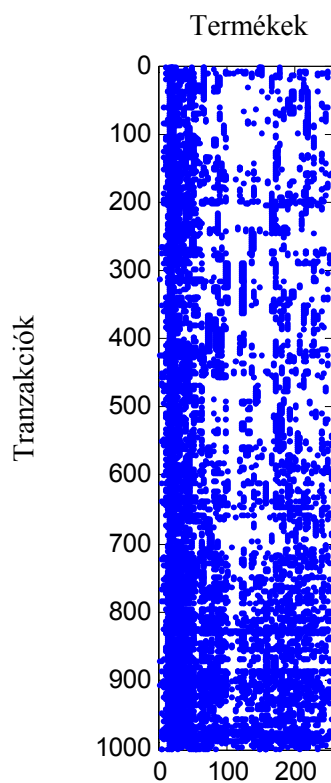


5.7. ábra: Az elemhalmazok VAT-ja

Egy másik, rövid példán pedig egy webáruház kereskedelme látható:

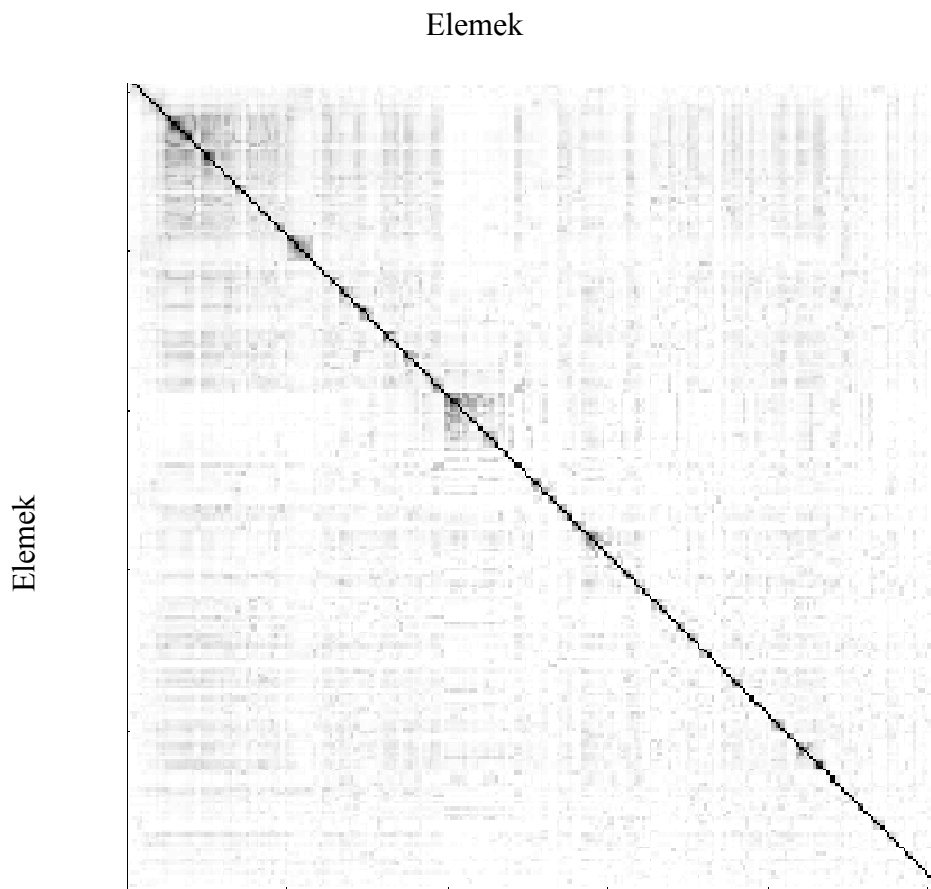


5.8. ábra: Az eredeti adatsorok



5.9. ábra: A biclusterezéssel csoportosított adatok

Az eredmény a webáruház vásárlóinak és termékeinek csoportosítása lesz.



A fenti ábra jól mutatja az elemek hasonlóságát a szerint rendezve, hogy azok mennyire hasonló módon látogattak.

### 5.3. Mérnöki alkalmazás

A dolgozatunk során többször is emlegetett gazdasági szempontokon kívül az általunk kifejlesztett algoritmusok más szakterületeken is hasznosíthatóak, mint például mérnöki alkalmazásoknál. A generált megoldások a folyamatbányászatban manapság aktívan kutatott biclustering eszközök fejlesztésének irányába mutatnak, a létrehozott döntéstámogató eszköz pedig felhasználható diszkrét eseménysorok elemzésére.

Egy termelőrendszerben sok ilyen diszkrét esemény merül fel a folyamatok során, és amennyiben a termékek, illetve vásárlói kosarak helyett az inputnak ezeket az eseményeket tekintjük, az output azon események csoportjai lesznek, melyek együtt gyakran állnak fenn. Számos termeléssel, vagy éppen vegyészettel, kutatás-fejlesztéssel foglalkozó vállalatnál ismertek olyan események vagy események sorozatai, melyek bekövetkezése kockázatot jelent. Ezek feltárása komoly szakértői ismeretet kíván, és nagyobb kihívást jelent. Programunk ilyen irányú alkalmazásával kinyerhetőek a gyakran együtt álló történések, melyekből a tapasztalt mérnökök különböző következtetéseket vonhatnak le, és akár az incidens-menedzsment probléma felügyeleti stádiumában hatékonyan felhasználhatnak. (Ebben a stádiumban a bekövetkezett hibák okát kutatják.) Amennyiben vannak cut-set-ek, és/vagy path-set-ek, az asszociációs szabályok gyorsabban felállíthatók, a meghibásodás gyanújára vagy a meghibásodásra a reakcióidő kisebb lehet. (A cut-set-ek azon események, amelyek közül, ha bármelyik is bekövetkezik, akkor a meghibásodás fennáll, a path-set-ekbe pedig olyan események tartoznak, amelyek közül egy fennállása is elég ahhoz, hogy a hiba biztosan ne következzen be.) A dolgozat során létrehozott algoritmussal megkönnyíthetjük ezen kritikus események feltárását.

További terület lehet a bioinformatikai felhasználás, ahol a tranzakcióknak gének sorozatai vannak megfeleltetve. A különböző génekre képzett gyakori elemhalmazokkal nem csak betegségek, de akár betegségekre való hajlamok is kimutathatóak, ezzel is segítve az orvostudomány munkáját.

A felsorolt két alkalmazáson kívül még számos más módja van a dolgozatunkban leírt algoritmus hasznosításának, ezzel is szemlélítve, mennyire sokrétűek és korszerűek a különböző adatbányászati módszerek.

## 6. Összefoglalás

Az adatbányászat egyik alapkövei a gyakori elemhalmazok. A velük való dolgozás, számolás és a feltárásuk nagyban elősegíthetik a vállalatok stratégiájának optimális alakítását, és akár a profit növekedéséhez is hozzájárulhatnak. A gyakori elem feltáró algoritmusok általánosságuknál fogva rengeteg helyzetben megállják a helyüket, jól alkalmazhatóak különféle problémákra szinte bármely területen. A dolgozatunkban bemutatott gyakori elemhalmaz kereső algoritmusok az Apriori-elvet megfelelően ötvözik a mátrixműveletekkel és bittable-technikákkal. A Visual Assessment of (Cluster) Tendency (VAT) algoritmus, és a Biclustering a gyakori elemhalmaz keresésén való ábrázolásban hatékonyan alkalmazhatóak, és egy új megközelítésű megoldást nyújtanak. Úgy érezzük, dolgozatunkkal elértük a kitűzött céljainkat, és egy könnyen interpretálható és implementálható programot sikerült létrehoznunk, amely hatékonyan találja meg, és megfelelően szemlélteti a gyakori elemhalmazokat. A bittable algoritmus továbbgondolásából sikerült egy új, gyorsabb algoritmust felállítanunk, amely a probléma megoldását egy újabb szemszögből közelíti meg. A programunk a mérnöki gyakorlatban is jól használható, akár nagyobb rendszerbe való implementálással, akár független felhasználásként is. A feladat továbbgondolásával, és a program specializálásával számos folyamatnál hasznos segédeszköz lehet, de önmagában is megállja a helyét, mint döntéstámogató eszköz. Úgy véljük, a munkánk során létrejövő szellemi termékek hozzájárulnak az adatbányászat előrehaladásához, fejlődéséhez; és a használt algoritmusokat a tudományág még fel fogja tudni használni az innovatív fejlődés érdekében.

## 7. Irodalomjegyzék

- Abonyi János, 2006.: Adatbányászat, a hatékonyság eszköze / ComputerBooks
- Akdes Serin, Martin Vingron, 2011.: DeBi: Discovering Differentially Expressed Biclusters using a Frequent Itemset Approach/ Algorithms for Molecular Biology, Volume 6, Issue 1, Pages 18.
- Anbarasi, D. Sathya Srinivas, Dr. K. Vivekanandan, 2011: Dimension Reduction Techniques for Market Basket Analysis. 2011 International conference on Information and Electronics Engineering, Singapore
- Balambér Dávid, 2011: Gyakori elemhalmazok kinyerése. Budapesti Műszaki és Gazdaságtudományi Egyetem, Villamosmérnöki és Informatikai Kar
- Bodon Ferenc, 2010.: Adatbányászati algoritmusok / Budapesti Műszaki és Gazdaságtudományi Egyetem
- Cristina Davino, Dario Bruzzese, 2003.: Visual post-analysis of association rules / Journal of Visual Languages & Computing, Volume 14, Issue 6, Pages 621–635.
- David Hand, Heikki Manilla, Padhraic Smyth., 2001: Principles of Data Mining / The MIT Press
- Hans-Peter Kriegel, Daniel A. Keim, 1996.: Visualization techniques for mining large databases: A comparison/ IEEE Transactions on knowledge and data engineering, Volume 8, No. 6, Pages 923-938.
- J. Gu, J. S. Liu, 2008: Bayesian biclustering of gene expression data. BMC Genomics
- J.C. Bezdek, R.J. Hathaway, 2002.: VAT: a tool for visual assessment of (cluster) tendency / Conference Publications: Neural Networks, 2002. IJCNN '02. Proceedings of the 2002 International Joint Conference on, Volume 3, Pages 2225 – 2230.
- Jiadong Ren, Qian Wang, Meng Wang, Shasha Feng, 2011.: An Algorithm based on Equal – Length Bit Table for Mining Frequent Patterns in Data Streams / AISS:

Advances in Information Sciences and Service Sciences, Volume 3, No. 7, pages 244 – 250.

Jiawei Han and Micheline Kamber, 2006.: Data Mining: Concepts and Techniques, 2nd ed./ Morgan Kaufmann Publishers

Jiawei Han, 2004.: Adatbányászat – koncepciók és technikák / Panem kft.

Li Yang, 2003: Visual exploration of association rules/ Proceedings of the IEEE Conference on Visualization Vis, Volume: 3, Pages: 1994-1995

Mátyus Péter. 2005: Gyakori elemhalmazok és asszociációs szabályok megjelenítése Sammon leképezéssel /TDK dolgozat/. Pannon Egyetem, Folyamatmérnöki Tanszék, Veszprém

Michael J. Berry, Gordon Linoff, 1997.: Data Mining Techniques: Marketing, Sales and Customer Support / John Wiley & Sons, Inc.

Pach Ferenc Péter. 2007: Rule-Based Knowledge Discovery for Process Data Analysis /Doktori (Ph.D.) értekezés/. Pannon Egyetem, Vegyészmérnöki Tudományok Doktori Iskola, Veszprém

Pang-Ning Tan, Michael Steinbach, Vipin Kumar, 2006.: Introduction to Data Mining / Addison-Wesley Companion Book Site

S. Mitra, H. Banka, 2006: Multi-objective evolutionary biclustering of gene expression data. Pattern Recognition, Kolkata

Yasemin Boztuğa, Thomas Reuttererb, 2008: A combined approach for segment-specific marketbasket analysis/ European Journal of Operational Research, Volume 187, Issue 1, Pages 294–312.

Z. Zhang, A. Teo, B. C. Ooi, K.-L. Tan, 2004: Mining Deterministic Biclusters in Gene Expression Data. Fourth IEEE Symposium on Bioinformatics and Bioengineering, Taichung, Singapore, Pages 283-290

## 8. Mellékletek

### 8.1. A továbbfejlesztett bittable technika pszeudokódja

*Input:*

$\sigma$  = support (gyakoriságküszöb)

$T$  = transactions (tranzakciók)

$P$  = products (termékek)

$N$  = size( $T$ ) (tranzakciók száma)

$n$  = size( $P$ ) (termékek száma)

*Output:*

$Bitems\{k\}$  = megfelelő támogatottsági szintet elért termékek

$Bi\{k\}$  = a support megfelelő termékek a  $B$  mátrixból

```

1: %Preprocessing:
2:   B=[]
3:   for every transaction  $T_i$  do
4:       for every product  $P_j$  do
5:           if  $T_i$  included  $P_j$  then  $B_{ij}=1$ 
6:           else  $B_{ij}=0$ 
7:           end if
8:       end for
9:   end for
11:

```

```

12:  %Wondermatrix technique
13:  bitb=B*BT
14:  for i=1 to N do
15:      for j=1 to N do
16:          if i≤j then bitbij=0
17:          end if
18:      end for
19:  end for
20: Bmax=max(bitb)
21:  for k=1 to Bmax do
22:      [dum1, dum2]= where bitbij ≥ k
23:      for every [dum1, dum2] [dum1, dum2]i do
24:          if IT{k} not included [dum1, dum2]i then IT{k} +=
              [dum1, dum2]i
25:          end if
26:      end for
27:  end for
28: %Here we analyse the wondermatrix technique
29:  for every product Pj do
30:      for every transaction Ti do
31:          tempk+=Bij
32:      end for
33:      if tempk ≥ σ then Bitems{1}=tempk
34:      end if
35:  end for

```

---

```

36: Bi{1}= Bitems{1}

37:   for k=2 to Bmax do

38:       x= where sum(abs( differences between Bitems{k-1}
           from 1 to end-1 )) not = 0

39:       index= [0, x ,size(Bitems{k-1})]

40:       for i=1: size(index-1) do

41:           v=[from index(i)+1 to index(i+1)]

42:           m=Bi{k-1}(IT{k},v)' * Bi{k-1}(IT{k},v)

43:           [dum1,dum2]= where m>=supp

44:           for j=size(dum1) do

45:               Bitems{k}=[Bitems{k},Bitems{k-1}(v(dum1j)),
                           Bitems{k-1}(v(dum2j)) ]

46:               Bi{k}=[Bi{k}, Bi{k-1}(v(dum1j)*v(dum2j))

47:           end for

48:       end for

49:       sort(Bitems{k})

50:       sort(Bi{k})

51:   end for

```

**8.2. A különböző elemek kiválogatása**

```
1: termék=[];

2: for i=1:1:length(kosarak)

3:     sor=kosarak(i,:);

4:     for j=1:1:length(sor)

5:         bennevan=0;

6:         for k=1:1:length(termék)

7:             if strcmp(termék(k),sor(j))==1

8:                 bennevan=1;

9:             end

10:        end

11:        if (bennevan==0)&(~isnan(cell2mat(sor(j))))

12:            termék=[termék sor(j)];

13:        end

14:    end

15:end
```

### 8.3. Bináris mátrixba való konvertálás

```
1: bin_matrix=[];
2:     for i=1:1:N,
3:         A=kosarak(i,:);
4:         temp=[];
5:         for k=1:n
6:             bennevan=0;
7:             for j=1:1:length(A)
8:                 if strcmp(A(j),termek(k))==1
9:                     bennevan=1;
10:                end
11:            end
12:            if bennevan==0
13:                temp=[temp,0];
14:            else
15:                temp=[temp,1];
16:            end
17:        end
18:        bin_matrix=[bin_matrix;temp];
19:    end
```

**8.4. Bittable technika alkalmazása**

```

1: [N,n]=size(A);

2: items{1}=find(sum(A,1)>=suppp)';

3: k=1; Ai{1}=A(:,items{1});

4: while ~isempty(items{k})

5:     k=k+1; Ai{k}=[]; items{k}=[];

6:     index=[0; find(sum(abs(diff((items{k-1})(:,1:end-
        1)))),(2)~=0); size(items{k-1},1)];

7:     for i=1:length(index)-1

8:         v=[index(i)+1:index(i+1)]; m=Ai{k-1}(:,v)'*Ai{k-
        1}(:,v);

9:         m=triu(m,1); [dum1,dum2]=find((m)>=suppp);

10:        for j=1:size(dum1,1)

11:            items{k}=[items{k}; [items{k-1}(v(dum1(j))),:]
            items{k-1}(v(dum2(j)),end) ] ];

12:            Ai{k}= [ Ai{k} Ai{k-1}(:,v(dum1(j)))].*Ai{k-
            1}(:,v(dum2(j)))];

13:        end

14:    end

15:    [items{k},I]=sortrows(items{k}); Ai{k}=Ai{k}(:,I);

16: end

```

**8.5. A VAT algoritmus használata**

```

1: function order=VAT(PairDist)

2: %% Prim's minimal spanning tree algorithm with special
   initialization

3: % initialization

4: [maxr maxc] = find(PairDist == max(max(PairDist)));

5: inic = maxr(1);

6: inic2 = maxr(2);

7: % iteration

8: order = [inic];

9: dum = [];

10: for i = 2 : size(PairDist,1)

11:     % + new sample

12:     dum = [dum; PairDist(order(end), :)];

13:     dum(:,order(end)) = inf;

14:     dum(end, order) = inf;

15:     % find the minimum distance

16:     [minr minc] = find( dum == min(min(dum)) );

17:     order = [order; minc(1)]; %order based on the similarity

18: end

19:

20: %the pairwise distances (REORDEING)

21: D=(PairDist);

22: D=D(order,:);

23: D=D(:,order);

```

```
24: surf(flipud(D), 'EdgeColor', 'none')

25: grid off

26: %mesh(flipud(D))

27: view([0 0 90])

28: colormap('gray')

29: axis equal

30: % for VAT graylevel

31: a = get(gcf, 'ColorMap');

32: a = a * ( 1-min(min(D))/max(max(D)) ) +
    min(min(D))/max(max(D));

33: colormap(a);

34: set(gca, 'YTickLabel', []);

35: set(gca, 'XTickLabel', []);

36: axis([0 size(PairDist,1) 0 size(PairDist,1)])

37: title('VAT on the original data')

38: hold off
```