# EVALUASI KINERJA ALGORITMA *ASSOCIATION RULE*

(*Performance Evaluation of Association Rule Algorithms*)

**GYSBER JAN TAMAELA**
*Jurusan Matematika FMIPA UNPATTI*
Jl. Ir. M. Putuhena, Kampus Unpatti, Poka-Ambon
E~Mail: gjtamaela@yahoo.com

**ABSTRACT**

Association is a technique in data mining used to identify the relationship between itemsets in a database (association rule). Some researches in association rule since the invention of AIS algorithm in 1993 have yielded several new algorithms. Some of those used artificial datasets (IBM) and claimed by the authors to have a reliable performance in finding maximal frequent itemset. But these datasets have a different characteristics from real world dataset. The goal of this research is to compare the performance of Apriori and Cut Both Ways (CBW) algorithms using 3 real world datasets. We used small and large values of minimum support thresholds as atreatment for each algorithm and datasets. As a result we find that the characteristics of datasets have a signifcant effect on the performance of Apriori and CBW. Support counting strategy, horizontal counting, showed a better performance compared to vertical intersection although candidate frequent itemsets counted was fewer.

***Keyword:*** *Association rule, Apriori, Cut Both Ways, maximal frequent itemset*

## PENDAHULUAN

Salah satu dari aplikasi *data mining* yang sangat penting adalah *association rule* yang diperkenalkan oleh Agrawal *et al.* (1993). Penelitian tentang *association rule* tidak hanya diutamakan pada penemuan algoritma-algoritma baru, tetapi juga perbandingan terhadap algoritma-algoritma yang ada dan belum pernah diperbandingkan sebelumnya, misalnya Zheng *et al.* (2001) dan ulasan tentang algoritma-algoritma dalam *association rule*, seperti Mueller (1995), Rantzau (1997), dan Dunham *et al*.( 2004).

Perbandingan-perbandingan yang dilakukan dapat diklasifikasikan dalam beberapa aspek, tetapi secara umum, kinerja (*performance*) dari sebuah algoritma *association rule* dikatakan baik, jika waktu eksekusi (respon waktu) yang dihasilkan adalah minimal. Menurut Zheng *et al*. (2001), jika ditinjau dari gugus data yang dipakai, para penemu algoritma-algoritma *association rule* menggunakan gugus data buatan yang dibuat oleh IBM (IBM *Artificial*). Gugus data ini dibuat sedemikian sehingga diharapkan dapat merepresentasikan gugus data penjualan dari perusahaan *retail* (Agrawal & Srikant, 1994). Tetapi dalam kenyataannya, karakteristik dari gugus data yang dihasilkan tidak sesuai dengan karakteristik data yang sebenarnya (*real world datasets*). Zheng, Kohavi dan Mason (2001) melakukan perbandingan algoritma-algoritma dalam *association rule* dengan menggunakan *real world data* dan gugus data IBM *Artificial*, dan menghasilkan kesimpulan bahwa adalah kurang tepat untuk membandingan algoritma *association rule* dengan menggunakan gugus data IBM *Artificial*. Sehingga dalam penelitian ini, penulis menggunakan gugus data *retail*.

### Tujuan

Tujuan penelitian ini adalah:
Membandingkan kinerja algoritma *CBW* dan *Apriori* terhadap gugus data *retail* (*real world dataset*).

### Ruang lingkup

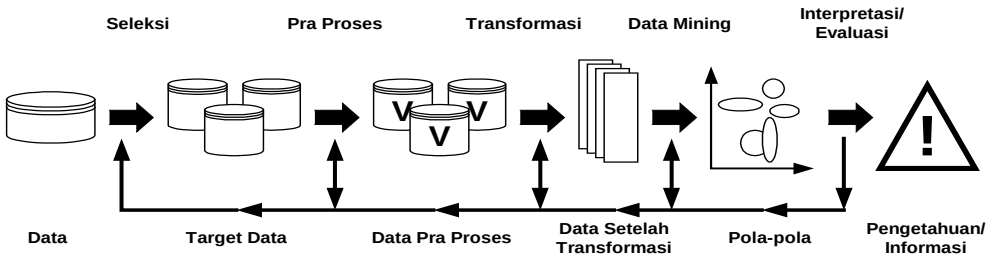
Untuk membatasi ruang lingkup pengkajian, penulis melakukan pembatasan sebagai berikut:

1. Gugus data yang dipakai dalam penelitian ini adalah data transaksi penjualan barang dari Swalayan Sinar Prima Bogor dari tanggal 1 Maret 2004 sampai dengan 21 Mei 2004 yang telah melewati proses seleksi sampai dengan transformasi dalam tahapan *KDD*.
2. Parameter pembanding adalah respon waktu masing-masing algoritma dalam menemukan *maximal frequent itemset*

## TINJAUAN PUSTAKA

### *Data Mining*

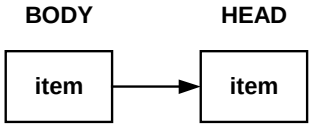
Terminologi *Knowledge Discovery in Databases* (*KDD*) menggambarkan proses untuk mengidentifikasi ide baru dan pola yang secara potensial berguna dari data yang tersimpan dalam basis data. Proses penemuan pengetahuan (*knowledge discovering process*) ini memiliki beberapa tahapan yang interaktif dan iteratif. Dari antara beberapa tahapan tersebut adalah aplikasi algoritma untuk mengekstraksi pola-pola dari data, yang disebut *data mining* – oleh sebab itu ada yang menyamakan *data mining* dengan *Knowledge Discovery in Database* (*KDD*) (Mueller 1995). Secara grafis, proses *KDD* digambarkan oleh Rantzau (1997) dapat dilihat dalam Gambar 1.




**Gambar 1.** Proses *Knowledge Discovery in Database* (Rantzau 1997)

***Association Rule***

*Association rule* dapat digambarkan sebagai hubungan antara *item-item* di mana *item* pada bagian anteseden dikelompokkan sebagai BODY dan *item* pada bagian konsekuen sebagai HEAD, seperti pada Gambar 2.




**Gambar 2.** Deskripsi *Association Rule*

*Association rule* digunakan untuk mengidentifikasikan relasi antara *itemset* dalam sebuah basis data. Relasi ini tidak berdasar pada sifat yang melekat (*inherent properties*) dari data tersebut seperti ketergantungan fungsional, tetapi lebih kepada kejadian bersama (*co-occurrence*) dari *item-item* pada data. Ukuran yang digunakan dalam menentukan apakah sebuah *itemset* merupakan *association rule* adalah jika *itemset* tersebut memenuhi ambang batas minimum ***support*** dan ***confidence*** yang berturut-turut disebut ***minsupp*** dan ***mincof***.

Menurut Agrawal dan Srikant (1993) ada 2 tahapan untuk menentukan *association rule* dari sebuah data yaitu:

1. menemukan *maximal frequent itemsets*
2. membangkitkan *association rule* dari *maximal frequent itemsets* yang didapat dari langkah ke-1.

Beberapa algoritma ada yang membatasi sampai pada menentukan maximal frequent itemset saja tanpa menemukan association rule seperti algoritma Pincer Search (Lin & Kedem 2002).

(Agrawal *et al*. 1993) dan (Cheung *et al*. 1996) memberikan beberapa definisi yang berkaitan dengan *association rule*.

**Definisi 1**: Misalkan $I = \{I_1, I_2, \ldots, I_m\}$ adalah sebuah gugus dari *m* atribut yang berbeda, disebut juga *literal*. *D* adalah basis data, di mana setiap *record* (*tuple*) *t* memiliki pengidentifikasi yang unik, dan mengandung sebuah *itemset* sedemikian hingga $t \subseteq I$. Sebuah *association rule* adalah sebuah implikasi dari bentuk $X \Rightarrow Y$, dimana $X, Y \subset I$ adalah gugusan dari *item-item* yang disebut *itemset*, dan $X \cap Y = \varnothing$. *X* disebut anteseden dan *Y* disebut konsekuen.

**Definisi 2**: ***Support*** dari *association rule* $X \Rightarrow Y$ adalah rasio dari *record* yang mengandung $X \cup Y$ dengan total *record* dalam basis data. Secara matematis dapat ditulis,

$$supp(X \Rightarrow Y) = \frac{\|t \in D \mid X \cup Y \subseteq t\|}{\|D\|} \quad \ldots\ldots\ldots\ldots(1)$$

Sementara ***minsupp*** menandakan ambang batas (*threshold*) yang menentukan apakah sebuah *itemset* akan dipergunakan pada perhitungan selanjutnya untuk menemukan *association rule*.

**Definisi 3**: ***Confidence*** dari *association rule* $X \Rightarrow Y$ adalah rasio dari *record* yang mengandung $X \cup Y$ dengan total *record* yang mengandung *X*. Secara matematis dapat ditulis,

$$conf(X \Rightarrow Y) = \frac{\|t \in D \mid X \cup Y \subseteq t\|}{\|t \in D \mid X \subseteq t\|} \quad \ldots\ldots\ldots(2)$$

Sementara ***mincof*** menandakan ambang batas dari sebuah *itemset* untuk menjadi *maximal frequent itemset*.

**Strategi penghitungan *support***

Menurut Su dan Lin (2004), sejauh ini baru terdapat 2 metode untuk menghitung *support* dari setiap *candidate itemset*, yaitu: *horizontal counting* dan *vertical intersection*.

Strategi *horizontal counting* menentukan nilai *support* dari setiap *candidate itemset* dengan melakukan *scan* terhadap transaksi dalam basis data satu persatu dan menambahkan *counter* dari *candidate itemset*, jika *candidate itemset* tersebut merupakan himpunan bagian (*subset*) dari transaksi. Sementara *vertical intersection* bekerja pada saat format basis data direpresentasikan secara vertikal sedemikian hingga setiap *record* berasosiasi dengan sebuah *item* untuk menyimpan pengenal dari setiap transaksi yang mengandung *item* tersebut, yang disebut *TIDlist*.

Algoritma-algoritma yang menggunakan strategi *horizontal counting* antara lain: *Apriori, FP-growth, Top-down*, dan *DIC*. Sementara yang menggunakan strategi *vertical intersection* antara lain: *Partition* dan *Eclat.* Algoritma *CBW* sendiri menggabungkan kedua strategi di atas.

**Algoritma *Cut Both Ways (CBW)***

Algoritma *CBW* ini ditemukan oleh Ja-Hwung Su dan Wen-Lin dari Universitas I-Shou, Taiwan pada tahun 2004. Sistem arah pencarian *maximal frequent itemset* yang diadopsi oleh algoritma ini adalah *hybrid*-2 (*frequent*-$\alpha$ *itemset to top* dan *frequent*-$\alpha$ *itemset to bottom*), dimana $\alpha$ = *cutting level*.

**Definisi 4**: Misalkan *D* adalah tabel transaksi dan $t_i$ adalah transaksi ke-*i*, maka:

$$\alpha = INT\left[\frac{\Sigma\left|t_{i\perp minsup}\right|}{\left|D\right|}\right]$$

dimana INT[$r$] menandakan pembulatan dari nilai $r$, untuk $r \geq 1$, dan $t_{i\perp minsupp}$ adalah gugus dari *item-item* dalam $t_i$ dengan *support* lebih dari *minsupp*, atau secara spesifik,

$$t_{i\perp minsupp} = \{\, x \mid x \in t_i \text{ dan } supp(x) \geq minsupp\}$$

Algoritma *CBW* terdiri dari sebuah algoritma utama yang akan mengeksekusi 3 buah prosedur yaitu: prosedur *Trans*, prosedur *DownSearch* dan prosedur *UpSearch*.

**Algoritma Utama *CBW***

**Input**: Basis data Transaksi ($D$) dan nilai *minimum support* (*minsupp*);
**Output**: Gugus *maximal frequent itemset* ($F$);
1. scan $D$ untuk membangkitkan semua frequent 1-*itemset*s in $F_1$;
2. *Trans*($D$, $T$, $F_1$, $F_2$, $\alpha$);
3. *Dwnsearch*($D$, $DF$, $F_\alpha$, $\alpha$, *minsupp*);
4. *Upsearch*($T$, $UF$, $F_\alpha$, $\alpha$, *minsupp*);
5. **return** $F = DF \cup UF$;

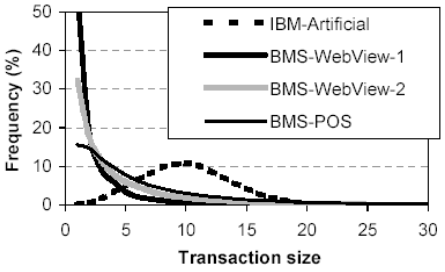
**Gugus data**

Beberapa penelitian tentang *association rule* lebih menitikberatkan pada penggunaan gugus data IBM *Artificial*, kalaupun menggunakan data *artificial* dan *real world dataset*, karakteristik data dari kedua jenis gugus data tersebut tidak pernah dibahas. Seperti dalam (Brin *et al*. 1997) yang menggunakan gugus data sensus di Amerika, (Su dan Lin 2004) yang menggunakan gugus data *retail*. Agrawal dan Srikant (1994) adalah yang menemukan dan menggunakan gugus data *artificial* pertama kali. Gugus data *artificial* ini dibuat sedemikian rupa sehingga menyerupai gugus data *retail* (Agrawal dan Srikant 1994).

Baru pada (Zheng *et.al.* 2001), penelitian ditiitikberatkan pada efek dari perbedaan karakteristik gugus data pada kinerja dari algoritma *association rule*. Hasil yang didapatkan dalam penelitian ini, adalah bahwa jika pada gugus data *artificial* kinerja sebuah algoritma *association rule* lebih baik dari algoritma lain, maka tidak serta merta berlaku juga pada *real world dataset*. Dalam penelitian yang dilakukan Zheng *et al.*, gugus data yang digunakan selain 1 buah gugus data *artificial*, juga 1 buah gugus data *electronic retail*, dan 2 buah gugus data *e-commerce*.

Zheng *et.al.* (2001) menemukan perbedaan dalam distribusi ukuran transaksi seperti pada Gambar 3.

Dari Gambar 3, terlihat bahwa karakteristik dari gugus data IBM *Artificial* sangatlah berbeda dengan *real world data*. Pada gugus data IBM *Artificial*, diasumsikan transaksi terbanyak terjadi pada 30% dari ukuran maksimum transaksi, sementara pada kenyataannya ukuran transaksi dengan frekuensi terbanyak terjadi pada ukuran-ukuran transaksi kecil.




**Gambar 3.** Distribusi transaksi gugus data IBM *Artificial* dan *real world dataset* (Zheng *et.al*., 2001)

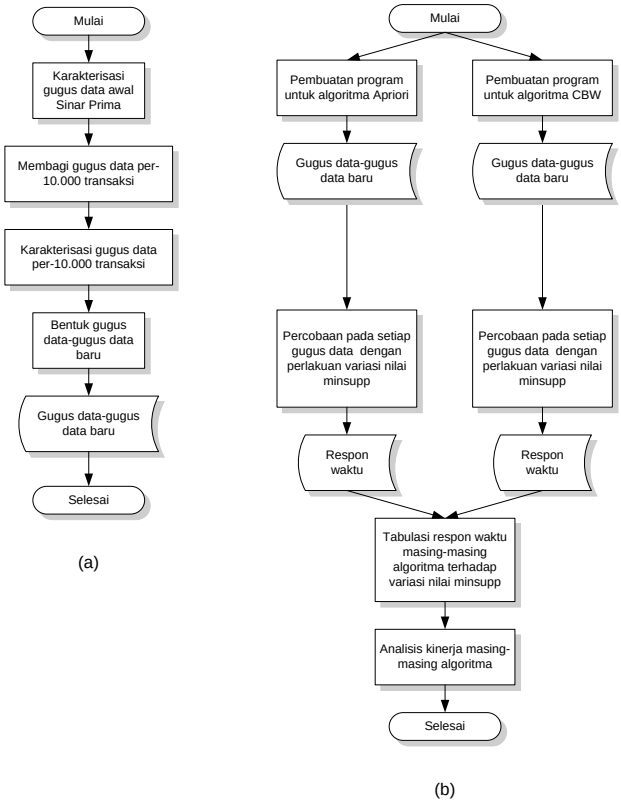
## BAHAN DAN METODE

**Bahan**

Bahan dalam penelitian ini adalah data transaksi penjualan barang Swalayan Sinar Prima Bogor dari tanggal 1 Maret 2004 sampai dengan 21 Mei 2004 yang telah melalui tahapan transformasi dalam proses *KDD*.

**Metode**

Metode yang digunakan dalam penelitian ini adalah bagian dari metode *KDD* yaitu proses *data mining*, selengkapnya dapat dilihat dalam Gambar 4.




**Gambar 4.** (a) pra proses data (b) tahapan percobaan

**Percobaan**

Masing-masing algoritma akan dibuat programnya dengan menggunakan spesifikasi perangkat lunak dan perangkat keras yang sama yaitu, Perangkat lunak: Microsoft Windows XP, Microsoft Excel 2003, Matlab versi 6.5 *release* 13; Perangkat Keras: *Processor* AMD

Athlon XP 1600+, 256 MB DDR RAM, *Harddisk* 40 GB (7200 rpm), *Mouse* dan *Keyboard*, Monitor

Pada percobaan yang dilakukan, perlakuan yang diberikan kepada masing-masing algoritma sama yaitu *minimum support*.

Ukuran *minsupp* yang digunakan mengacu pada (Zheng *et al*. 2001), selengkapnya dalam Tabel 1.

**Tabel 1.** Ukuran *minsupp* yang dipakai dalam percobaan

| Kategori | Ukuran Minsupp (%) |
|---|---|
| *Minsupp* besar | 0.2, 0.4, 0.6, 0.8, 1.0 |
| *Minsupp* kecil | 0.02, 0.04, 0.06, 0.08, 0.10 |

Setiap satuan percobaan dilakukan pengulangan sebanyak 3 kali, untuk mendapatkan nilai yang akurat, terutama pada nilai respon waktu yang sangat kecil. Dalam melaksanakan percobaan, komputer tidak dalam keadaan melakukan kegiatan yang lain.

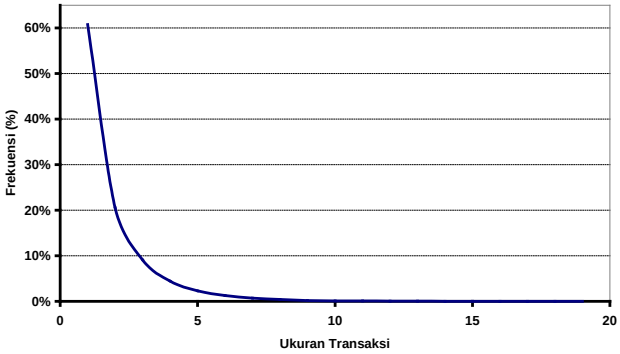
## HASIL DAN PEMBAHASAN

### Karakteristik umum gugus data

Gugus data yang dipergunakan untuk membandingkan algoritma *Apriori* dan *CBW* ini adalah gugus data *boolean* yang telah melewati proses Transformasi dalam tahapan *KDD*. Karakteristik dari gugus data penjualan Swalayan Sinar Prima adalah: ∑ Transaksi = 70.898, ∑ *Item* = 3.652, ∑ Kategori *Item* = 35, Ukuran Transaksi Maksimum = 19, Rata-rata Transaksi = 1,79.

Berdasarkan sumber dan karakteristiknya, selanjutnya penulis memakai nama **SPαI35D70K** pada gugus data ini. Dimana ***I*** menandakan jumlah *item* dan ***D*** menggambarkan jumlah *record*. Dalam penelitian ini, penulis tidak menggunakan jumlah 3.652 merek dagang sebagai *item*, tetapi kategori *item* yang berjumlah 35 buah. Pengelompokan *item* berdasarkan kategorinya dengan cara membentuk taksonomi *item* untuk menemukan *association rule* ini disebut *generalized association rule* (Srikant & Agrawal, 1995). Taksonomi adalah hirarki *is-a* dari *item-item* yang memungkinkan untuk menemukan asosiasi antara setiap level dalam hirarki (Rantzau, 1997). Keuntungan dari menggunakan taksonomi ini adalah menghasilkan jumlah *item* yang lebih sedikit, sehingga menghasilkan kombinasi *itemset* yang sedikit juga juga. Ini tentunya menghemat waktu eksekusi.

Gambar 5 menunjukkan bahwa distribusi ukuran transaksi dari SPαI35D70K menyerupai distribusi ukuran transaksi dari *real world data* dari Zheng *et.al* (2001).




**Gambar 5.** Distribusi ukuran transaksi SPαI35D70K

Untuk kepentingan analisis pada tahapan perbandingan algoritma, SPαI35D70K akan dipecah-pecah menjadi beberapa gugus data yang baru. Bertolak dari sini, penulis membagi SPαI35D70K menjadi 3 buah gugus data baru yaitu: SPαI10D50K, SPαI15D50K dan SPαI20D50K..

Karakterisitik dari ketiga gugus data-gugus data baru yang diperoleh dari gugus data SPαI35D70K tergambar dalam tabel di bawah ini.

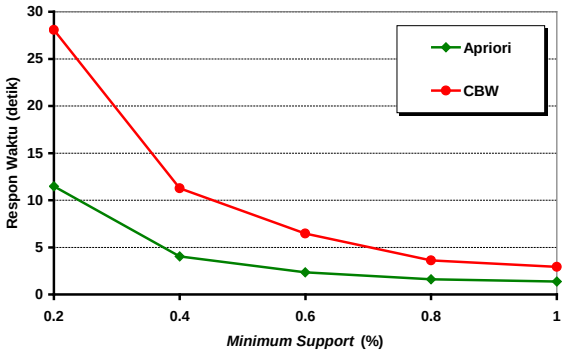
**Tabel 2.** Karakteristik dari gugus data turunan SPαI35D70K

| Gugus Data | \| *I* \| | \| *D* \| | *Size* | Max. Transaksi | Rata-rata Transaksi |
|---|---|---|---|---|---|
| **SPαI20D50K** | 20 | 54.586 | 8,200 MB | 14 | 1,65 |
| **SPαI15D50K** | 15 | 49.535 | 5,979 MB | 12 | 1,62 |
| **SPαI10D50K** | 10 | 45.642 | 5,061 MB | 10 | 1,50 |

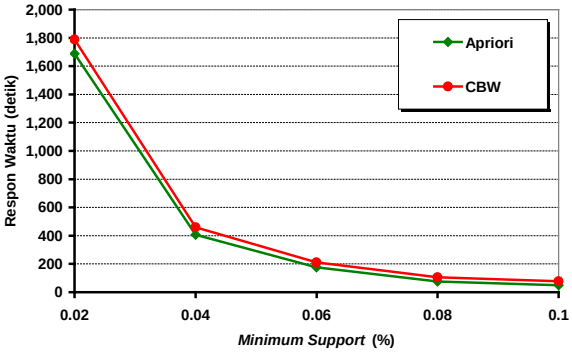
### Hasil eksperimen

Pada eksperimen ini yang menjadi parameter pengukuran adalah kecepatan respon waktu terhadap perlakuan nilai *minimum support (minsupp)* yang berbeda. Algoritma dikatakan berhenti dan dihitung respon waktunya jika telah menemukan *maximal frequent itemset*.

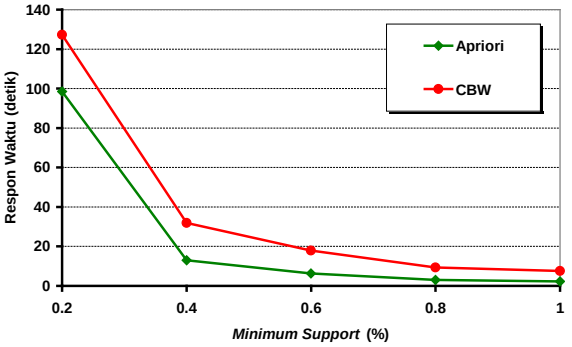
Dalam Gambar 6 sampai dengan Gambar 11, disajikan grafik hasil eksperimen terhadap gugus data-gugus data. Tiap gambar dikelompokan berdasarkan kategori *minsupp*.



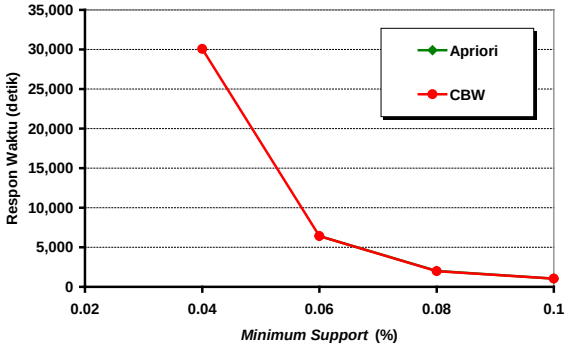

**Gambar 6.** Respon waktu gugus data SPαI10D50K untuk *minsupp* besar



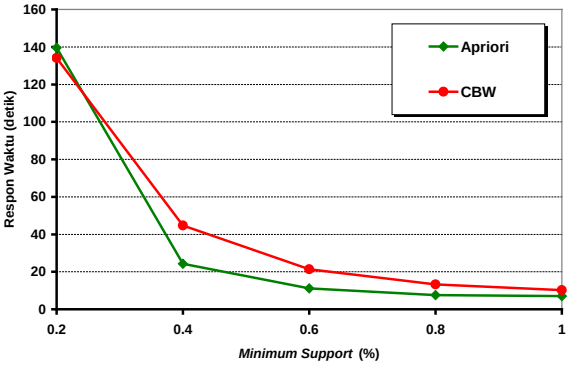

**Gambar 7.** Respon waktu gugus data SPαI10D50K untuk *minsupp* kecil



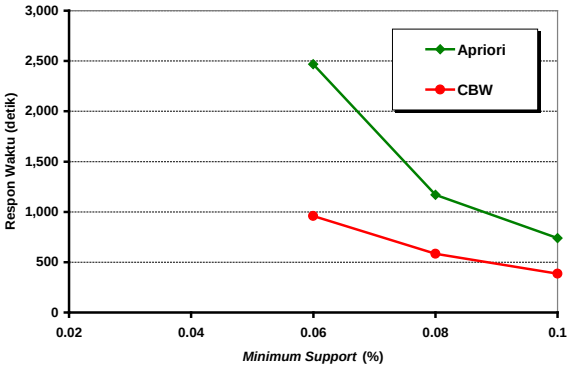

**Gambar 8.** Respon waktu gugus data SPαI15D50K untuk *minsupp* besar




**Gambar 9.** Respon waktu gugus data SPαI15D50K untuk *minsupp* kecil




**Gambar 10.** Respon waktu gugus data SPαI20D50K untuk *minsupp* besar




**Gambar 11.** Respon waktu gugus data SPαI20D50K untuk *minsupp* kecil

Dari grafik-grafik di atas, maka dapat dilihat bahwa:

1. Untuk gugus data SPαI10D50K, pada semua nilai *minsupp* , algoritma *Apriori* memiliki respon waktu yang selalu lebih baik dari algoritma *CBW* untuk setiap nilai *minsupp*. Selisih respon waktu *Apriori* dan *CBW* yang terkecil terdapat pada nilai *minsupp* yang paling besar yaitu 1%, atau sekitar 1 detik dan selisih terbesar terdapat pada nilai *minsupp* 0.2% yaitu sekitar 100 detik. Sementara untuk nilai *minsupp* yang kecil, *Apriori* juga memiliki respon waktu yang sangat jauh lebih baik dari *CBW* bahkan untuk semua nilai *minsupp*. Jika kita lihat *trend*-nya, maka jika nilai *minsupp* semakin kecil, maka selisih respon waktu algoritma *Apriori* dan *CBW* akan semakin besar dan nilainya menjadi semakin signifikan.
2. Pada gugus data SPαI15D50K, untuk nilai *minsupp* besar, *Apriori* masih mengungguli *CBW*. Tapi pada saat nilai *minsupp* kecil *CBW* lebih cepat dari *Apriori*. Algoritma *CBW* tidak dapat menyelesaikan eksperimen pada nilai *minsupp* 0.02%. Bahkan algoritma *Apriori* hanya mampu menyelesaikan sampai pada nilai *minsupp* 0.06% saja. Hal yang menarik yaitu pada gugus data inilah algoritma *CBW* mulai bisa mengatasi kecepatan dari algoritma *Apriori*, terutama pada nilai *minsupp* 0.04% dimana *Apriori* tidak mampu menyelesaikan kurang dari 10 jam, sementara *CBW* menyelesaikannya dalam waktu 8,5 jam. Jika dilihat *trend*-nya, maka untuk gugus data ini, semakin kecil nilai *minsupp*, maka selisih respon waktu dengan *CBW* juga menjadi semakin signifikan.
3. Pada gugus data SPαI20D50K, respon waktu *CBW* masih lebih lama dari *Apriori*, kecuali pada saat mencapai nilai *minsupp* = 0.2%, dimana *CBW* unggul dari *Apriori* sekitar 5 detik. Sementara untuk nilai *minsupp* kecil, kedua algoritma tidak mampu menyelesaikan kurang dari 10 jam untuk nilai *minsupp* 0.04% dan 0.02%. Secara keseluruhan, algoritma *CBW* selesai lebih cepat dalam menemukan *maximal frequent itemset*, bahkan pada nilai *minsupp* 0.06% selisih respon waktu algoritma *Apriori* dan *CBW* mencapai sekitar 25 menit. Jika dilihat *trend*-nya, maka untuk gugus data ini, semakin kecil

*minsupp*, maka selisih respon waktu *Apriori* dan *CBW* menjadi semakin signifikan.

Secara keseluruhan, hasil eksperimen terhadap ketiga gugus data yang memiliki jumlah *item* yang berbeda terangkum pada Tabel 3. Dapat dilihat bahwa keunggulan sebuah algoritma terhadap algoritma yang lain tidak bisa terpelihara untuk jumlah *item* yang berbeda.

**Tabel 3.** Rangkuman hasil eksperimen terhadap SPαI10D50K, SPαI15D50K, dan SPαI20D50K

| Gugus data | *Minsupp* besar | *Minsupp* kecil |
|---|---|---|
| SPαI10D50K | APR > CBW | APR > CBW |
| SPαI15D50K | APR > CBW | CBW > APR |
| SPαI20D50K | APR > CBW | CBW > APR |

Ket: Posisi lebih kiri menunjukkan kinerja yang lebih baik

**Analisis hasil eksperimen**

Pada bagian ini penulis akan mencoba melakukan analisis dari hasil eksperimen yang telah ditampilkan. Analisis akan dilakukan terhadap beberapa parameter yang memberi pengaruh terhadap kecepatan algoritma *association rule* dalam menemukan *maximal frequent itemset*. Parameter-parameter adalah: jumlah *item*, jumlah *record* dan jumlah *scan/pass* terhadap basis data, arah pencarian *maximal frequent itemsets* dan strategi penghitungan *support*.

**Pengaruh jumlah *item***

Banyaknya *item* dalam basis data memiliki pengaruh yang sangat besar dalam metode *association rule*, terutama untuk gugus data *retail*, dimana *item* yang akan diolah berjumlah banyak. Jumlah *itemset* akan bertambah secara eksponensial menurut rumus $2^p - 1$, dimana $p$ menunjukkan banyaknya *item*. Untuk mengolah data dengan jumlah *item* sebanyak 20 buah saja, maka jumlah *itemset* yang akan muncul adalah 1.048.575 *itemset*. Walaupun dalam kenyataannya, sangat kecil sekali kemungkinan kedudukan *maximal frequent itemset* adalah gugus *item* yang ke-1.048.575 tadi.

Pada saat jumlah *item* sedikit maka *Apriori* menunjukkan kinerja yang terbaiknya, tetapi keunggulan *Apriori* tadi mulai tereduksi seiring dengan bertambahnya jumlah *item*, yang berarti bertambahnya jumlah kombinasi *itemset*. Dapat dilihat juga keunggulan masing-masing algoritma terhadap gugus data dengan jumlah *item* yang berbeda selalu berubah-ubah.

**Arah pencarian *maximal frequent itemset***

Su dan Lin (2004), menggambarkan bahwa arah pencarian yang digunakan dalam algoritma *CBW* adalah *hybrid* yaitu *top-down* dari $F_\alpha$ menuju $F_3$, dan *bottom-up*, $F_{\alpha+1}$ menuju $F_n$. Tetapi dalam eksperimen yang dilakukan, pencarian yang dilakukan hanya *bottom-up*. Hal ini disebabkan karena nilai α-*cut* yang dihasilkan kurang dari 3 untuk ketiga gugus data, tidak seperti yang diprediksi oleh Hwang Su dan Yang Lin, bahwa nilai α-*cut* minimal adalah 3.

**Tabel 4.** Nilai α setiap gugus data untuk masing-masing nilai *minsupp*

| Gugus Data | *Minsupp* (%) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 0.02 | 0.04 | 0.06 | 0.08 | 0.10 | 0.20 | 0.40 | 0.60 | 0.80 | 1.00 |
| SPαI10D50K | 1.495 | 1.495 | 1.495 | 1.495 | 1.495 | 1.495 | 1.495 | 1.495 | 1.495 | 1.495 |
| SPαI15D50K | 1.623 | 1.623 | 1.623 | 1.623 | 1.623 | 1.623 | 1.623 | 1.617 | 1.617 | 1.617 |
| SPαI20D50K | 1.646 | 1.646 | 1.646 | 1.646 | 1.646 | 1.646 | 1.646 | 1.637 | 1.637 | 1.637 |

Dalam (Su & Lin 2004), pemberian nilai 3 sebagai minimal α-*cut* adalah kesimpulan dari eksperimen terhadap gugus data buatan IBM yang dipakai. Hal ini sesuai dengan karakteristik data yang telah dibahas di atas, bahwa kinerja algoritma terhadap gugus data artifisial dan *real world dataset* sangat berbeda. Sementara algoritma *Apriori* adalah algoritma yang menggunakan arah pencarian *bottom-up*.

Perbedaannya dalam kasus ini, dalam melakukan penghitungan *support* dari masing-masing *itemset*, *Apriori* dan *CBW* menggunakan strategi yang berbeda. *Apriori* menggunakan *horizontal counting*, sementara *CBW* menggunakan *vertical intersection*. Di lain pihak Jumlah *scan/pass* terhadap basis data hanya dapat dihitung pada algoritma yang mengadopsi strategi *horizontal counting*. Oleh sebab itu, jumlah *scan/pass* pada *CBW* tidak dapat dihitung.

**Strategi penghitungan *support***

Seperti yang telah dikemukakan di atas, algoritma *Apriori* dan *CBW* menggunakan strategi penghitungan *support* yang berbeda. Strategi *horizontal counting* menghitung nilai *support* dari masing-masing *itemset* dengan cara melakukan *scan* terhadap basis data satu persatu. Su dan Lin (2004) menyimpulkan bahwa pendekatan ini baik jika jumlah kandidat *frequent itemset* yang dihasilkan sangat sedikit, tetapi jika sebaliknya jumlah kandidat *frequent itemset* yang dihasilkan banyak, maka waktu yang dibutuhkan untuk menghitung kandidat pada level berikutnya semakin banyak.

Sementara pada *CBW* jumlah kandidat *frequent itemset* yang dihasilkan lebih sedikit dibandingkan dengan jumlah kandidat *frequent itemset* yang dihasilkan oleh *Apriori* pada semua gugus data, seperti tergambar dalam Tabel 5 sampai dengan Tabel 7.

**Tabel 5.** Perbandingan jumlah *frequent itemset* dan jumlah kandidat *frequent itemset* yang dihitung algoritma *Apriori* dan *CBW* pada gugus data SPαI10D50K

| ***Minsupp* (%)** | **Jumlah *Frequent Itemset*** | ***Apriori*** | ***CBW*** |
|---|---|---|---|
| 0.02 | 601 | 645 | 601 |
| 0.04 | 409 | 491 | 409 |
| 0.06 | 329 | 410 | 329 |
| 0.08 | 266 | 365 | 266 |
| 0.10 | 233 | 328 | 233 |
| 0.20 | 140 | 212 | 142 |
| 0.40 | 86 | 149 | 92 |
| 0.60 | 59 | 110 | 72 |
| 0.80 | 44 | 88 | 64 |
| 1.00 | 38 | 79 | 61 |

**Tabel 6.** Perbandingan jumlah *frequent itemset* dan jumlah kandidat f*requent itemset* yang dihitung algoritma *Apriori* dan *CBW* pada gugus data SPαI15D50K

| ***Minsupp* (%)** | **Jumlah *Frequent Itemset*** | ***Apriori*** | ***CBW*** |
|---|---|---|---|
| 0.02 | - | - | - |
| 0.04 | 1089 | 1544 | 1093 |
| 0.06 | 767 | 1168 | 773 |
| 0.08 | 589 | 990 | 596 |
| 0.10 | 489 | 815 | 502 |
| 0.20 | 257 | 479 | 283 |
| 0.40 | 135 | 296 | 177 |
| 0.60 | 90 | 210 | 132 |
| 0.80 | 62 | 162 | 117 |
| 1.00 | 54 | 152 | 112 |

**Tabel 7.** Perbandingan jumlah *frequent itemset* dan jumlah kandidat *frequent itemset* yang dihitung algoritma *Apriori* dan *CBW* pada gugus data SPαI20D50K

| ***Minsupp* (%)** | **Jumlah *Frequent Itemset*** | ***Apriori*** | ***CBW*** |
|---|---|---|---|
| 0.02 | - | - | - |
| 0.04 | - | - | - |
| 0.06 | 960 | 1744 | 987 |
| 0.08 | 726 | 1390 | 765 |
| 0.10 | 573 | 1128 | 623 |
| 0.20 | 284 | 650 | 364 |
| 0.40 | 139 | 377 | 257 |
| 0.60 | 93 | 275 | 193 |
| 0.80 | 67 | 226 | 182 |
| 1.00 | 55 | 209 | 179 |

Dalam tabel-tabel di atas jumlah kandidat *frequent itemset* yang memiliki nilai *support* tidak sama dengan nol untuk *Apriori* selalu lebih banyak dari *CBW*. Hal ini disebabkan karena dalam Prosedur *Trans* pada algoritma *CBW*, banyak sekali *item* yang dibuang, sehingga jumlah kandidat *frequent itemset* telah tereduksi sebelum melakukan *vertical intersection*. Jika dalam gugus data SPαI10D50K kinerja *CBW* tidak lebih baik dari *Apriori*, hal ini dikarenakan waktu dibutuhkan untuk melakukan *bit vector intersection* masih lebih banyak walaupun jumlah kandidat *frequent itemset* lebih sedikit. Tetapi dalam gugus data SPαI15D50K dan SPαI20D50K, jumlah kandidat *frequent itemset* yang dibangkitkan oleh algoritma *CBW* tereduksi sampai setengah jumlah yang dibangkitkan *Apriori*, sehingga mampu mereduksi waktu eksekusi dari algoritma *CBW*.

**Kompleksitas algoritma**

*Running time* dari masing-masing algoritma adalah sebagai berikut:

$$Apriori \quad \Rightarrow \quad T = |D|.\left\{ S_{H_1}.C_1^n + \sum_{i=2}^{n} S_{H_i}.C_i^{P_i} \right\}$$

$$CBW \quad \Rightarrow \quad T = |D|.\left\{ \sum_{i=1}^{2} S_{H_1}.C_1^{P_i} + \sum_{i=3}^{\alpha} S_{Hi}.C_i^{Q_i} + \sum_{i=\alpha}^{n} S_{V_i}.C_i^{R_i} \right\}$$

; dimana $P_i \le Q_i \le R_i$, $P_1 = n$

Keterangan:
- $S_{Hi}$ = *running time* untuk penghitungan *Support Horizontal counting* pada *i- itemset*
- $|D|$ = jumlah *record* dalam basis data
- $n$ = jumlah *item*
- $C_a^b$ = kombinasi *a item* dari *b item*
- $P_i$ = jumlah *item* pada *i-itemset*
- $r_i$ = banyaknya *record* dalam tiap partisi
- $\alpha$ = level ke-*i* dimana *CBW* mempartisi basis data
- $Q_i$ = jumlah *item* pada *frequent i-itemset* prosedur *Downsearch* algoritma *CBW*
- $S_{Vi}$ = *running time* untuk penghitungan *Support Vertical intersection* pada *i-itemset*
- $R_i$ = jumlah *item* pada *frequent i-itemset* prosedur *Upsearch* algoritma *CBW*

Kondisi pada skenario kasus terburuk (*worst case scenario*) adalah jika *maximal frequent itemset* adalah *longest frequent itemset* yang berarti setiap algoritma akan menelusuri setiap kombinasi dari jumlah *item*. Implikasinya adalah $P = Q = R = n$, dan $r.m = |D|$. Sementara kasus terbaik (*best case scenario*) terjadi bila tidak ada satupun *itemset* yang merupakan *frequent itemset*. Kompleksitas dari masing-masing algoritma untuk kondisi terbaik dan terburuk terdapat pada Tabel 8.

**Tabel 8.** Kompleksitas masing-masing algoritma untuk skenario kasus terbaik dan terburuk

| **Algoritma** | ***Best case*** | ***Worst case*** |
|---|---|---|
| *Apriori* | $O(\|D\|.S_{H_1}.n)$ | $O\left(\|D\|.\sum_{i=1}^{n} S_{H_i}.C_i^n\right) = O(\|D\|.2^n)$ |
| *CBW* | $O(\|D\|.S_{H_1}.n)$ | $O\left(\|D\|.\left\{\sum_{i=1}^{\alpha} S_{Hi}.C_i^n + \sum_{i=\alpha}^{n} S_{V_i}.C_i^n\right\}\right)$ |

## KESIMPULAN

Dari penelitian yang telah dilakukan, maka beberapa kesimpulan yang dapat diambil yaitu:

1. Karakteristik data sangat berpengaruh pada kinerja dari algoritma-algoritma dalam *association rule*, termasuk *Apriori* dan *CBW*.
2. Algoritma *Apriori* menunjukkan kinerja yang lebih baik pada saat *minsupp* besar, sementara *CBW* menunjukkan kinerja yang lebih baik pada saat *minsupp* kecil
3. Algoritma *CBW* tidak dapat menunjukkan kelebihannya dalam melakukan strategi pencarian *maximal frequent itemset* secara *hybrid* pada gugus data yang digunakan, karena nilai *α-cut* yang

dihasilkan tidak seperti yang diperkirakan penemu algoritma ini. Hal ini mengakibatkan prosedur *DownSearch* tidak dapat dilakukan, dan *CBW* hanya melakukan strategi pencarian *bottom-up*.

4. Rasio jumlah kandidat *frequent itemset* yang dibangkitkan oleh algoritma *CBW* dibandingkan dengan algoritma *Apriori* semakin besar seiring dengan bertambahnya jumlah *item*. Hal ini mengakibatkan *respon* waktu yang diihasilkan oleh *CBW* lebih kecil dari *Apriori*
5. Analisis terhadap jumlah *scan/pass* tidak dapat dilakukan karena *CBW* menggunakan strategi penghitungan support yang berbeda.
6. Strategi penghitungan *support*, *horizontal counting*, menunjukkan kinerja yang lebih baik dibandingkan dengan *vertical intersection* walaupun jumlah kandidat *frequent itemset* yang dibangkitkan lebih sedikit.

## DAFTAR PUSTAKA


Agrawal R, Imielinski T, Swami A. 1993. Mining Association Rules between Sets of *Item*s in Large Database, Proceedings of 1993 ACM SIGMOD Conference, Washington DC, USA

Agrawal R, Srikant R. 1994. Fast Algorithms for Mining Association Rules, Research Report RJ 9839, IBM Almaden Research Center, San Jose, California

Cheung DW, Vincent TN., Ada WC., Yongjian F. 1996, Efficient Mining of Association Rules in Distributed Databases, IEEE Transactions on Knowledge and Data Engineering, Vol 8, No. 6, pp. 911-922

Dunham MH, Xiao Y, Gruenwald L, Hossain Z . 2004, A Survey of Association Rules, http://www.cs.uh.edu /~ceick/6340/grue-assoc.pdf [12 Juli 2004]

Han JW dan Kember. 2001. Data mining concepts and techniques. Simon Fraser University Academic Press, Micheline, USA

Lin D, Kedem ZM 2002, Pincer Search: An Efficient Algorithm for Discovering the Maximum Frequent Set, IEEE Transaction on Knowledge and Data Engineering, Vol. 14 No. 3, pp. 553 – 566.

Mueller A. 1995, Fast Sequential and Parallel Algorithms for Association Rule Mining: A Comparisson, http://www.hpc.isti.cnr.it/~palmeri/datam/articles/cs/ tr/3515.ps

Park JS, MS Chen, PS Yu. 1995. An Effective Hash-Based Algorithm for Mining Association Rules. Proceeding of ACM SIGMOD International Confrence on Management of Data. San Jose, CA, USA: 175-186

Rantzau R. 1997, Extended Concepts for Association Rule Discovery, http://www.cs.bme.hu/~bodon/kozos/datamining/valaszthato/rantzau97-extended.pdf

Su JH, Lin WY. 2004, CBW: An efficient algorithm for frequent itemset mining, Proceeding of the 37th Hawaii International Conference on System Sciences.

Sucahyo YG. 2003, Data Mining: Menggali informasi yang terpendam, Artikel Populer Ilmu Komputer Copyright © 2003 http://www.ilmukomputer.com

Vallikona H. 2003. Association Rule Mining Over Multiple Database: Partitioned and Incremental Approach. Thesis. The University of Texas, Airlington.

Widodo S. 2004, Perancangan Data Mining dengan Metode Association Rule untuk Analisis Cross-Market (Studi Kasus Toko Sinar Bogor), Skripsi S1 Departemen Ilmu Komputer, FMIPA Institut Pertanian Bogor.

Zaki MJ. 2000, Scalable Algorithm for Association Mining, IEEE Transactionon Knowledge and Data Engineering, Vol. 12 No. 2, pp. 372 – 390.

Zheng Z, Kohavi R, Mason L. 2001, Real World Performance of Association Rule Algorithms, Proceeding of the 7th ACM-SIGKDD International Conference on Knowledge Discovery and Data Mining, New York.


## UCAPAN TERIMA KASIH

Bersama ini penulis ingin mengucapkan terima kasih yang sebesar-besarnya kepada Swalayan Sinar Prima Bogor, dalam hal ini Saudara Ibrahim, S.Kom yang telah membantu penulis dengan memberikan data penjualan Swalayan Sinar Prima Bogor untuk keperluan penelitian.