# PERBAIKAN ALGORITMA PENGGALIAN FREQUENT CLOSED ITEMSET CHARM

**Mardiyanto**[1] **Arif Djunaidy**[2]
[1]*Jurusan Teknik Informatika ITS, Surabaya, 60111, e-mail: mardiyanto@petrokimia-gresik.com*
[2]*Jurusan Teknik Informatika ITS, Surabaya, 60111, e-mail: arif@if-sby.edu*

**ABSTRAKSI**

*Penggalian frequent closed itemset merupakan salah satu bagian penting dari penggalian kaidah assosiasi (Association rule) karena dapat secara unik menentukan himpunan semua frequent itemsets dan supportnya.Berbagai algoritma penggalian frequent closed itemset telah ditemukan, diantaranya adalah algoritma CHARM dan algoritma DCI_CLOSED. Algoritma CHARM menggunakan format data vertikal diffset dan metode subsumption check untuk melakukan pemeriksaan duplikasi. Metode ini tidak efisien karena memerlukan penyimpanan semua frequent closed itemsets sebelumnya. Algoritma DCI_CLOSED menggunakan format data vertikal bitvectors dan menggunakan metode order preserving untuk melakukan pengecekan duplikasi. Metode ini efisien karena tidak memerlukan penyimpanan frequent closed itemsets sebelumnya.*

*Berdasarkan riset dan teori yang berkaitan dengan penggalian frequent closed itemsets, belum ada algoritma yang mengintegrasikan penggunaan format data vertikal diffset dan pengecekan duplikasi tanpa melakukan penyimpanan semua frequent closed itemsets sebelumnya. Sehingga ada peluang penelitian untuk merancang perbaikan algoritma CHARM yang lebih efisien penggunaan memorinya. Metodenya adalah menggabungkan subsumption check pada cabang yang sedang dienumerasi dan metode order preserving, sehingga tabel hash tidak menyimpan semua frequent closed itemsets sebelumnya.*

*Hasil penelitian menunjukkan algoritma perbaikan CHARM lebih efisien penggunaan memorinya bila dibandingkan dengan algoritma CHARM untuk nilai minimum support yang semakin kecil.*

***Kata kunci***: *Penggalian Kaidah Asosiasi,Perbaikan Algoritma CHARM, Frequent Closed Itemset, Diffset*

## 1. PENDAHULUAN

Penggalian *frequent closed itemset* merupakan satu bagian penting dari penggalian kaidah assosiasi (*Association rule*) karena dapat secara unik menentukan himpunan semua *frequent itemsets* dan *support*nya. Pada penggalian kaidah asosiasi, *closed sets* memiliki properti yang lebih lengkap bila dibandingkan dengan *maximal sets*. Telah banyak penelitian yang dilakukan untuk melakukan penggalian *frequent closed itemset* diantaranya CHARM dan DCI_CLOSED.

M.J.Zaki [4] dengan algoritma CHARM menggunakan format data vertikal diffset dan metode *subsumption check* untuk melakukan pemeriksaan duplikasi. *Subsumption check* tidak efisien karena harus menyimpan semua *frequent closed itemset* yang telah ditemukan sebelumnya.

C. Lucchesse, S. Orlando dan R. Perego [1] dengan algoritma DCI_CLOSED melakukan enumerasi *frequent closed itemset* menggunakan format data vertikal *bit vector* dan menggunakan metode *order preserving* untuk pemeriksaan duplikasi.

Kontribusi dari penelitian ini adalah tersedianya algoritma penggalian *frequent closed itemset* yang lebih efisien penggunaan memorinya bila dibandingkan dengan algoritma CHARM. Ide inovasi yang diterapkan dalam perbaikan algoritma adalah menggabungkan metode pemeriksaan duplikasi *order preserving* pada seluruh cabang dan *subsumption check* pada cabang yang sedang dienumarasi sehingga tidak perlu menyimpan seluruh *frequent closed itemset* yang ditemukan sebelumnya

## 2. BIDANG TERKAIT

Algoritma CHARM efisien untuk mengenumerasi himpunan semua *frequent closed itemsets* karena dengan menggunakan format data vertikal *diffset*[5] secara bersama-sama bisa memeriksa baik *itemset space* dan *transaction space* diatas *IT-tree* (*itemset tidset tree*) *search space*. Untuk menghapus *non closed itemsets* digunakan metode *subsumption check dengan* pendekatan *hash-based*

Kombinasi beberapa faktor ini membuat CHARM merupakan algoritma praktis dan efisien, namun demikian algoritma ini masih memiliki dua kelemahan [1]. Pertama, adanya komputasi *redundant* sehingga algoritma tidak dapat secara langsung melakukan penggalian *closed itemsets*. Kedua, penggunaan tabel *hash* untuk menyimpan semua *frequent closed itemsets* sebelumnya untuk pemeriksaan duplikasi tidak efisien karena *closed itemset* tumbuh secara eksponensial untuk minimum support yang makin kecil

DCI_CLOSED [1] mengenumerasi *frequent closed itemset* secara efisien dan cepat karena menggunakan format data vertikal *bitvectors* sehingga hanya dengan *bitwise and* interseksi dapat dilakukan. Pemeriksaan duplikasi dilakukan dengan metode *order preserving* yang sangat efisien penggunaan memorinya karena tidak perlu menyimpan *frequent closed itemset* sebelumnya.

Meskipun secara komputasi DCI_CLOSED cepat tetapi algoritma memiliki kelemahan karena menggunakan format data vertikal *bitvectors* yang mempunyai kompresi lebih rendah bila dibandingkan dengan format data vertikal *diffset* [5].

### 2.1 Penggalian Frequent Pattern

Misalkan $\chi$ adalah himpunan item dan D sebuah transaksi basis data, dimana setiap transaksi mempunyai sebuah identifikasi unik (*tid*) dan berisi himpunan item. Himpunan semua *tids* ditunjukkan sebagai $\tau$. Sebuah himpunan X $\subseteq \chi$ juga dinamakan *itemset* dan sebuah himpunan Y $\subseteq \tau$ dinamakan *tidset*. Sebuah *itemset* dengan k *items* dinamakan *k-itemset*. Itemset {A,C,W} ditulis sebagai ACW dan *tidset* {2,4,5} sebagai 245. *Itemset* X, hubungan *tidset*-nya ditunjukkan sebagai t(X), sedang untuk *tidset* Y, hubungan *itemset*-nya ditunjukkan sebagai I(Y). Sehingga hubungannya digambarkan t(X) = $\bigcap_{x \in X} t(x)$ dan i(Y) = $\bigcap_{y \in Y} i(y)$

t(ACW) = t(A) $\cap$ t(C) $\cap$ t(W)
= 1345 $\cap$ 123456 $\cap$ 12345 = 1345

dan i(12) = i(1) $\cap$ i(2) = ACTW $\cap$ CDW = CW. Digunakan notasi X x t(X) menunjukkan pasangan *itemset-tidset* dan disebut *IT-pair*.

*Support* [4] *itemset* X, ditunjukkan $\sigma$(X) = |t(X)| adalah jumlah transaksi yang terjadi pada sebuah *subset*. Sebuah itemset adalah *frequent* jika *support*nya ($\sigma$(X)) $\geq$ nilai *minimum support (min_sup*). *Frequent itemset* dinamakan *maximal* jika tidak ada subset itemset lainnya frequent. sebuah *frequent itemset* X dinamakan *closed* jika dan hanya jika c(X) = X [4]. Dengan kata lain, *frequent itemset* X dinamakan *closed* jika tidak ada *superset proper* Y $\supset$ X dengan $\sigma$(X) = $\sigma$(Y). Untuk *instance* pada Gambar 1.

c(AW) = i(t(AW))= i(1345) = ACW, *sehingga* AW tidak *closed*, sebaliknya c(ACW) = i(t(ACW))= i(1345)=ACW, sehingga ACW *closed.*

DISTINCT DATABASE ITEMS

| Jane Austen | Agatha Christie | Sir Arthur Conan Doyle | Mark Twain | P. G. Wodehouse |
|---|---|---|---|---|
| A | C | D | T | W |

DATABASE

| Transcation | Items |
|---|---|
| 1 | A C T W |
| 2 | C D W |
| 3 | A C T W |
| 4 | A C D W |
| 5 | A C D T W |
| 6 | C D T |

ALL FREQUENT ITEMSETS
MINIMUM SUPPORT = 50%

| Support | Itemsets |
|---|---|
| 100% (6) | C |
| 83% (5) | W, CW |
| 67% (4) | A, D, T, AC, AW CD, CT, ACW |
| 50% (3) | AT, DW, TW, ACT, ATW CDW, CTW, ACTW |

**Gambar 1**. Basis data Contoh

Dalam Gambar 1 ada 5 item yang berbeda, X = {A,C,D,T,W} dan 6 transaksi, $\chi$={1,2,3,4,5,6}. Gambar 1 menunjukkan 19 *frequent itemset* berisi paling sedikit tiga transaksi dengan *min_sup*=50%. Gambar 2 menunjukkan 7 *frequent closed itemset* yang terbentuk dari basis data contoh. Secara teoritis pada kondisi terburuk, ada $2^{|x|}$ *frequent* dan *frequent closed itemsets.*

### 2.2 Diffset Untuk Perhitungan Cepat Frekuensi

CHARM menggunakan format data vertikal *diffset*. Kelebihan utama menggunakan format data vertikal dibandingkan bila menggunakan format data horisontal adalah karena *support* dapat dihitung lebih mudah dan lebih cepat dengan melakukan interseksi pada *tidset* yang diperlukan dan ada pemangkasan otomatis informasi yang tidak relevan ketika proses interseksi.

Sebuah *class* dengan *prefix* P. d(X) menunjukkan *diffset* X. Dalam keadaan normal metode vertikal, ketersediaan *class* dengan *prefix* t(P) sama dengan ketersediaan semua anggota *class* t(PXi). Misalnya PX dan PY adalah sebarang dua anggota class P, dari definisi *support* t(PX) $\subseteq$ t(P) dan t(PY) $\subseteq$ t(P). Sehingga untuk memperoleh support PXY dengan meneliti kardinalitas dari

t(PX) $\cap$ t(PY) = t(PXY)

bila tersedia bukan t(PX) tetapi d(PX) = t(P) – t(X) yaitu perbedaan pada *tids* X dari P atau d(PY). Secara rekursif $\sigma$(PXY) dapat dihitung sebagai berikut:

$\sigma$(PXY) = $\sigma$(PX) - |d(PXY)|.

Untuk itu harus dihitung d(PXY). Dari definisi d(PXY) = t(PX) – t(PY) dengan hanya menggunakan *diffset* dengan persamaan berikut :

d(PXY) = t(PX) – t(PY) = t(PX) – t(PY) + t(P) – t(P)
= (t(P) – t(PY)) – (t(P) – t(PX))
= d(PY) – d(PX)

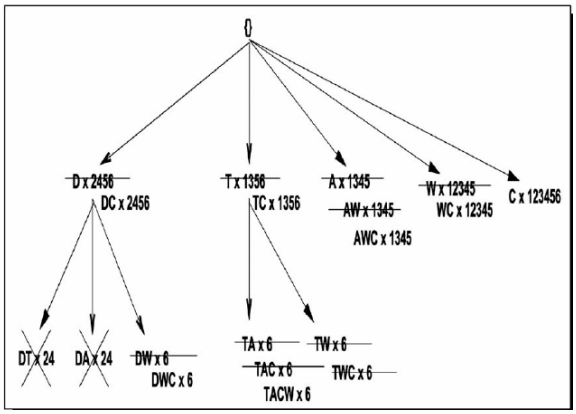
dengan kata lain, mengganti perhitungan d(XY) sebagai perbedaan tidset t(PX) – t(PY), dihitung sebagai perbedaan *diffset* d(PY) – d(PX).

Dengan menganggap basis data awal disimpan pada format data *tidset*, tetapi proses selanjutnya digunakan format data *diffset*. Misal m(Xi) dan m(Xj) menunjukkan jumlah ketidaksesuaian pada *diffset* d(Xi) dan d(Xj). Ada 4 kondisi untuk dipertimbangkan:

- Properti 1. m(Xi) = 0 dan m(Xj) = 0, lalu d(Xi) = d(Xj) atau t(Xi) = t(Xj)
- Properti 2. m(Xi) > 0 dan m(Xj) = 0, lalu d(Xi) $\supset$ d(Xj) atau t(Xi) $\subset$ t(Xj)
- Properti 3. m(Xi) = 0 dan m(Xj) > 0, lalu d(Xi) $\subset$ d(Xj) atau t(Xi) $\supset$ t(Xj)
- Properti 4. m(Xi) > 0 dan m(Xj) > 0, lalu d(Xi) $\neq$ d(Xj) atau t(Xi) $\neq$ t(Xj)

Gambar 2 menunjukkan pencarian *closed set* menggunakan *diffset* pengganti *tidset*. Penggalian persis sama dengan bila seandainya menggunakan format data vertikal tidset, tetapi saat ini dilakukan operasi perbedaan pada *diffset* (kecuali *class root*, menggunakan *tidset*). Misal sebuah *IT-pair* TAWC x 6, menunjukkan bahwa TAWC berbeda dari

*parent*-nya TC x 1356 hanya pada *tid* 6, dapat disimpulkan bahwa *IT-pair* sesungguhnya harus menjadi TAWC x 135.




**Gambar 2**. Proses Pencarian Menggunakan Diffset

## 3. PERBAIKAN ALGORITMA CHARM

Berdasarkan kelemahan yang dimiliki algoritma CHARM maka rancangan perbaikan algoritma yang dilakukan adalah:

a. Menggantikan proses SUBSUMPTION-CHECK dengan melakukan pemeriksaan *order preserving* [1] sehingga tidak diperlukan lagi tabel *hash* yang menyimpan seluruh *frequent closed itemset* yang sudah terbentuk.
b. Pemeriksaan *order preserving* harus dikombinasikan dengan *subsumption check* karena ada kemungkinan timbulnya *frequent itemset* yang tidak *closed* pada cabang yang sedang dienumerasi. Perbedaan mendasar adalah tabel *hash* yang dipergunakan hanya menyimpan *frequent closed itemset* pada cabang yang sedang dienumerasi.

Untuk itu dirancang perbaikan algoritma CHARM seperti terlihat pada Algoritma 1. Secara umum algoritmanya adalah sebagai berikut:

a. Inisialisasi *class*[P], simpul diperiksa untuk item tunggal *frequent* dan *tidset*-nya ($l_i$ x t($l_1$), $l_1 \in \chi$ ) pada baris 1. Dianggap bahwa elemen dalam [P] diurut sesuai dengan total orde *f* yang sesuai. Perhitungan utama dilakukan dalam CHARM-EXTEND yang menghasilkan himpunan *frequent closed itemset C*.
b. CHARM-EXTEND bertanggung jawab untuk mempertimbangkan kombinasi *IT-pair* $l_i$ x t($l_1$ ) yang datang setelahnya (baris 3) sesuai dengan jumlah orde f. Setiap $l_i$ membangkitkan sebuah prefix baru, Pi = P $\cup$ $l_i$ , dengan class [Pi], yang awalnya kosong(baris 4).
c. Pada baris 6, dua IT-pair dikombinasikan menghasilkan pair baru X x Y, dimana X = $l_j$ dan Y = t($l_i$ ) $\cap$ t($l_j$).
d. Baris 7 menguji empat properti *IT-pair* mana yang dapat diterapkan dengan memanggil CHARM-PROPERTY. Catatan bahwa rutin ini mungkin mengubah *class*[P] saat ini dengan menghapus *IT-pair* yang telah di *subsumed* oleh *pair* lainnya. Juga menambah IT-pair terbaru yang dibangkitkan pada *class*[Pi] baru. Juga dapat mengubah *prefix* Pi pada Properti 1 dan 2.
e. Lalu menambah *itemset* Pi dalam himpunan *closed itemsets* C (baris 9), menunjukkan bahwa Pi bukan *subsumed* oleh *closed set* yang ditemukan sebelumnya.
f. Sekali semua $l_j$ telah diproses, secara *recursive* mengeksplorasi class [Pi] baru dengan DFS (baris 10). Setelah kembali, semua *closed itemset* yang berisi Pi telah dibangkitkan. Lalu kembali ke baris 4 untuk melakukan proses berikutnya (*unpruned*) *IT-pair* pada [P}.

Pengurutan elemen dilakukan dengan melakukan sorting *itemset* didasarkan pada *support*-nya. Tujuannya adalah untuk meningkatkan peluang elemen yang dihapus dari *class*[P].

**Algoritma 1.** *Perbaikan Algoritma CHARM*

```
CHARM (D, min_sup):
1. [0]={li x t(li) :li ∈ χ ∧ σ(li)≥ min_sup}
2. CHARM-EXTEND ([0], C=ϕ)
CHARM-EXTEND([P], C):
3. for each li x t(li) in [P]
4.   Pi = P ∪ li dan [Pi] = ϕ
5.   for each lj x t(lj) in [P], with j > I
6.     X = lj dan Y = t(li) ∩ t(lj)
7.     CHARM-PROPERTY(XxY,li,lj,Pi,[Pi],[P])
8.   end for
9.   if !IS_DUP(Pi,Y)
10.    Write Out Pi and support
11.  CHARM-EXTEND ([Pi],C)
12.  delete [Pi]
13.  if  [P] > 1 and level=root then
14.    Preset = Preset ∪ (t(li))
15.    delete C
16.  end if
17.  delete li from [P]
18. end for
CHARM-PROPERTY(X x Y,li,lj,Pi,[Pi],[P]):
19. if ( σ(X) ≥ min_sup) then
20.   if t(li) = t(j) then // Properti 1
21.     delete lj from [P]
22.     Pi = Pi ∪ lj
23.   else if t(Xi) ⊂ t(Xj) then//Properti 2
24.     Pi = Pi ∪ lj
25.   else if t(Xi) ⊃ t(Xj) then//Properti 3
26.     delete lj from [P]
27.     Add X x Y to [Pi]
28.   else if t(Xi) ≠ t(Xj) then//Properti 4
29.     Add X x Y to [Pi]
30.   end if
31.  end if
IS_DUP(Pi,Y)
32. subset = false
33. h(Pi) = ∑_{T∈t(P)} T
34. for all Y ∈ HASHTABLE [h(Pi) do
35.    if σ(Y) ≠ σ(Pi) atau Pi ⊈ Y
36.        C = C ∪ Pi
37.    else
38.        subset=true
39.    end if
40. end for
41. if !subset then
42.   while all j ∈ PRESET & !subset
43.     if g(Pi) ⊆ g(j) then
44.         subset = true
45.     else
46.         subset = false
47.     end if
48.   end while
49. end if
50. return subset
```

Secara detail perbaikan yang dilakukan adalah sebagai berikut:

a. Baris 10, proses pemeriksaan duplikasi dengan menggunakan metode *order preserving* dan *subsubsumption* check pada cabang yang sedang dienumerasi, bila tidak ada duplikasi maka dilakukan *write out frequent closed itemset* dan *support*nya (baris 11).
b. Penambahanan preset dilakukan hanya pada *class root.* Setelah itu dilakukan penghapusan tabel *hash* dan li dari class [P] ( baris 13 – 17)

Pemeriksaan *order preserving generators* berdasar-kan *preset*nya. Bila diberikan sebuah generator *gen*= Y $\cup$ *i*, dimana Y adalah *closed itemsets* dan i $\notin$ Y, didefinisikan *pre-set*(*gen*) sebagai berikut :

$$Pre\text{-}set(gen)= \{ j \mid j \in \chi, j \notin gen, \text{dan } j \prec i \} \quad (1)$$

Untuk memeriksa properti *order preserving* dari *gen* dengan mempertimbangkan *tidlist g*(*j*), untuk semua $j \in$ *pre-set*(*gen*). Diberikan *gen* = Y $\cup$ *i* menjadi sebuah generator dimana Y adalah *closed itemsets* dan $i \notin$ Y. Jika $\exists j \in$ *pre-set*(*gen*) sedemikian sehingga $g(gen) \subseteq g(j)$ sehingga *gen* bukan *order preserving*. Jika sebuah generator bukan merupakan *order preserving* maka dapat dilakukan *pruning* terhadapnya sehingga dengan metode ini menjadi efisien karena tidak memerlukan penyimpanan *closed itemset* sebelumnya untuk melakukan pemeriksaan duplikasi.

## 4. ANALISIS HASIL PENGUJIAN

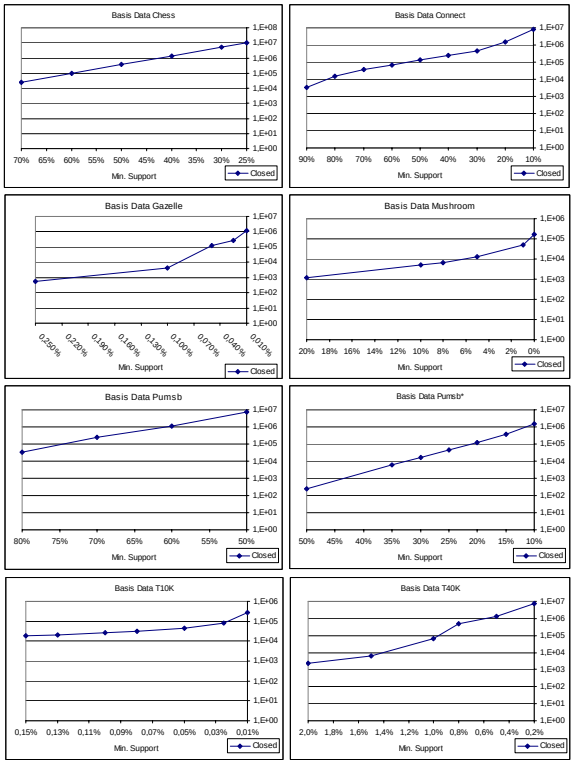
Pengujian dilakukan di PC Pentium IV 3000 Mhz, 1 GB RAM dengan sistem operasi Windows XP Pro. Algoritma dikodekan dengan Microsoft Visual C++ 6.0. Perbandingan dilakukan antara algoritma CHARM dan algoritma perbaikan CHARM (CHARM_NEW). Basis data dipilih dari beberapa basis data *real* dan *sintetik* yang diperoleh dari www.rcs.edu /~zaki/ sebagaimana dapat dilihat pada Tabel 1.

**Tabel** 1. Karakteristik Basis Data

| ***Basis Data*** | ***Jml Item*** | ***L*** | ***Jml Record*** | ***Max. Pattern*** | ***Min Sup*** |
|---|---|---|---|---|---|
| chess | 76 | 37 | 3196 | 21 | 25% |
| connect | 130 | 43 | 67557 | 29 | 10% |
| mushroom | 120 | 23 | 8124 | 21 | 0.075 % |
| pumsb* | 7117 | 50 | 49046 | 38 | 10% |
| pumsb | 7117 | 74 | 49046 | 22 | 60% |
| gazelle | 498 | 2.5 | 59601 | 154 | 0.01% |
| T10I4D100K | 1000 | 10 | 100000 | 13 | 0.01% |
| T40I10D100K | 1000 | 40 | 100000 | 20 | 0.2% |

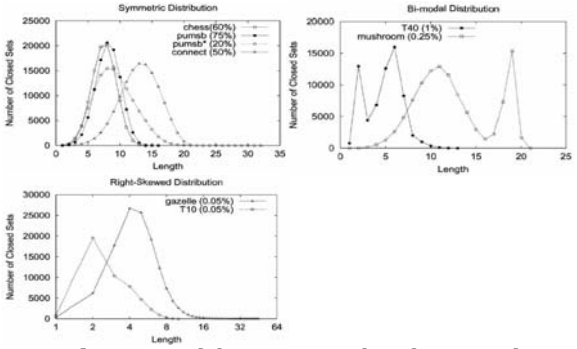
Basis data pumsb dan pumsb* berisi data sensus karakteristik berbagai spesies jamur. Pumsb* pada dasarnya sama dengan pumsb yang membedakan adalah pumsb* tidak mempunyai item dengan minimum *support* 80% atau lebih. Basis data connect dan chess diberikan dari tahapan game masing-masing. Tiga basis data berikutnya berasal dari UC Irvine Machine Learning Database Repository. Basis data sintetik (T10 dan T40) menggunakan IBM generator merupakan transaksi mimik pada lingkungan *retailing*. Basis data gazelle berasal dari *click-stream* data perusahaan dot-com kecil Gazelle.com yang tidak eksis lagi. Umumnya basis data *real* lebih rapat bila dibandingkan dengan basis data sintetik.

Tabel 1 menunjukkan karakteristik basis data real dan sintetik yang digunakan dalam evaluasi pada minimum *support* terendah yang digunakan pada pengujian. Gambar 3 menunjukkan total jumlah *frequent closed itemset* untuk berbagai macam basis data pada beberapa nilai *support*. Terlihat bahwa jumlah *frequent closed itemset* yang ditemukan tidak tergantung pada jumlah record yang dimiliki.




**Gambar 3**. Kardinalitas *Frequent Closed Itemset*

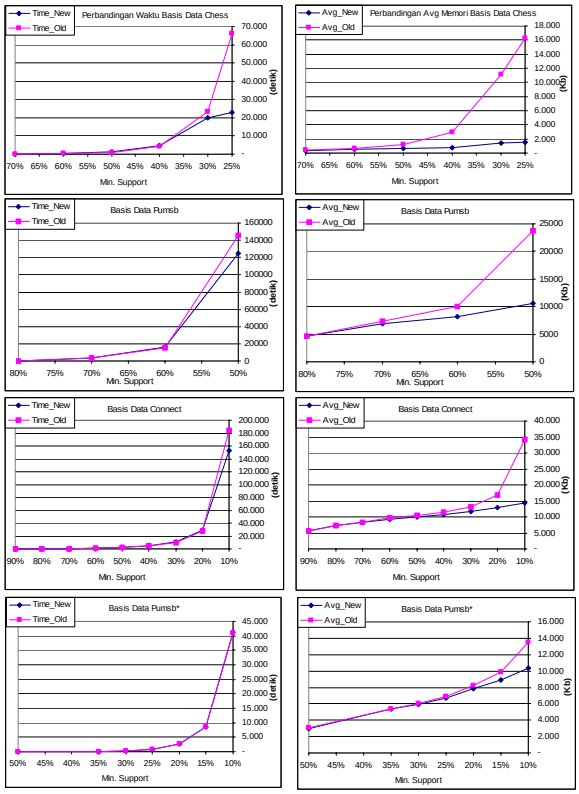
Basis data dikelompokkan menurut tipe distribusinya [4] sebagaimana terlihat pada Gambar 4. Chess, pumsb*, pumsb dan connect menunjukkan sebuah distribusi yang hampir simetrik dari *closed frequent pattern*. T40 dan mushroom menunjukkan kecenderungan distribusi *bimodal closed sets*. Sedangkan gazelle dan T10 mempunyai distribusi *right-skewed*.




**Gambar 4**. Jumlah *Frequent Closed Itemset* dan Distribusi Panjangnya [4]

### 4.1 Basis Data Simetrik

Basis data dengan distribusi *simetrik* adalah chess, pumsb, connect dan pumsb*. Sebagaimana dapat dilihat pada Gambar 5 untuk semua basis data simetrik memori yang digunakan algoritma CHARM_NEW lebih efisien bila dibandingan memori yang dipergunakan algoritma CHARM. Algoritma CHARM_NEW menunjukkan perbaikan yang cukup siginifikan karena bisa menghemat memori minimal 50% untuk support terkecil yang digunakan dalam pengujian. Kinerja terbaik ditunjukkan pada basis data chess dengan penghematan yang bisa dilakukan mencapai 87,6% hal ini disebabkan basis data chess jumlah recordnya relatif kecil (3196) sedangkan *frequent closed itemset* yang ditemukan bisa mencapai 5 juta untuk minimum support 30%.

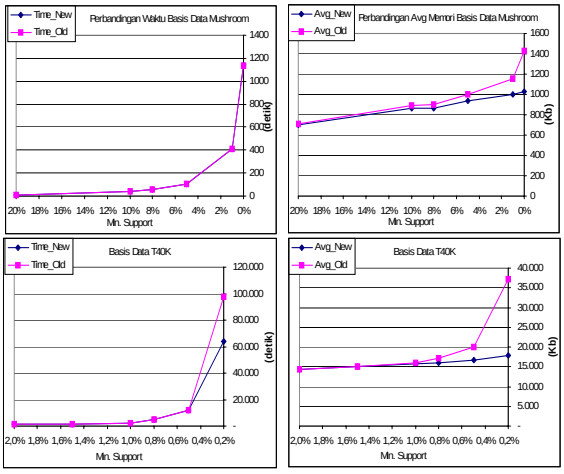



**Gambar 5**. Perbandingan Kinerja Algoritma CHARM_NEW pada basis data simetrik

Waktu proses yang digunakan pada basis data simetrik dapat dilihat pada Gambar 5, terlihat kinerja algoritma CHARM_NEW secara keseluruhan perbedaannya tidak terlalu signifikan. Untuk basis data chess waktu proses algoritma CHARM_NEW lebih baik bila dibandingkan dengan waktu proses algoritma CHARM untuk nilai support 25%, hal ini disebabkan pada minimum *support* tersebut jumlah *frequent closed itemset* yang ditemukan adalah lebih dari 10 juta. Pada proses ini algoritma CHARM memerlukan *virtual* memori yang lebih besar bila dibandingkan dengan algoritma CHARM_NEW, sehingga algoritma CHARM memerlukan waktu proses pembacaan data yang lebih lama.

### 4.2 Basis Data Bimodal

Dua basis data dengan distribusi bimodal yaitu mushroom dan T40 ditunjukkan pada Gambar 6. Algoritma CHARM_NEW menggunakan memori yang lebih efisien bila dibandingkan algoritma CHARM. Untuk basis data mushroom dengan jumlah record 8124 efisiensinya tidak begitu besar (maks 28%) hal ini disebabkan hasil enumerasi *frequent closed itemset* hanya 164.520 untuk minimum *support* 0,0075%, sehingga memori yang dipergunakan oleh algoritma CHARM untuk melakukan pemeriksaan duplikasi tidak begitu berbeda jauh dengan memori yang digunakan oleh algoritma CHARM_NEW yang berbasis *preset*. Sedangkan untuk basis data T40 penghematannya cukup signifikan sampai dengan 51,7% karena enumerasi *frequent closed itemset*nya bisa mencapai 7 juta untuk minimum *support* 0,2%.

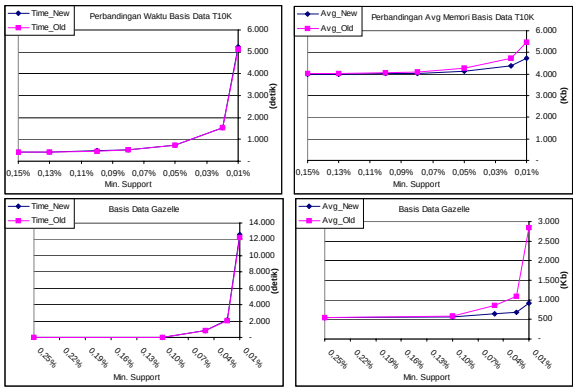



**Gambar 6**. Perbandingan Kinerja Algoritma CHARM_NEW pada Bimodal Data Set

Waktu proses secara keseluruhan perbedaannya tidak terlalu signifikan. Untuk basis data T40K waktu proses algoritma CHARM_NEW lebih baik bila dibandingkan dengan waktu proses algoritma CHARM untuk nilai minimum support 0,2%, hal ini disebabkan pada minimum support tersebut jumlah *frequent closed itemset* yang ditemukan adalah lebih dari 7 juta sehingga *virtual* memori algoritma CHARM jauh lebih besar daripada algoritma CHARM_NEW. Penggunaan *virtual* memori yang besar menyebabkan proses pembacaan menjadi lebih lama.

### 4.3 Basis Data Right Skewed

Pada basis data gazelle dan T10, mempunyai sejumlah besar *closed pattern* sangat pendek, kinerja kedua algoritma sebagaimana ditunjukkan pada Gambar 7. Untuk basis data gazelle, *frequent closed itemset* yang ditemukan bisa mencapai 1,2 juta untuk minimum support 0,01%. Efisiensi memori yang bisa dicapai algoritma CHARM_NEW bisa mencapai 68,1%, hal ini dikarenakan *closed pattern* basis data gazelle hanya ditemukan pada nilai minimum *support* yang yang sangat kecil ( $\leq$ 0,5%) sehingga *tidset* untuk semua *closed pattern* akan kecil ($\leq$ 298). Untuk basis data T10K efisiensi yang bisa dicapai jauh lebih kecil bila dibandingkan efisiensi yang bisa dicapai basis data Gazelle seperti ditunjukkan pada Gambar 8. Efisiensi memori yang bisa dicapai tidak terlalu besar hanya sebesar 13,5% untuk nilai minimum support 0,01%, hal ini dikarenakan pada nilai minimum support 0,01% jumlah *frequent closed itemset* yang ditemukan hanya sebesar 283 ribu berbeda jauh dengan basis data Gazelle yang bisa mencapai 1,2 juta.




**Gambar 7.** Perbandingan Kinerja Algoritma CHARM_NEW pada Right Skewed Data Set

Kinerja waktu proses enumerasi *frequent closed itemset* kedua algoritma ditunjukkan pada Gambar 8. Terlihat waktu proses algoritma CHARM_NEW secara keseluruhan perbedaannya tidak terlalu signifikan untuk berbagai macam nilai minimum support. Untuk semua nilai minimum *support* jumlah *frequent closed itemset* yang ditemukan tidak terlalu besar ( $\leq$ 1,3 juta) sehingga algoritma tidak memerlukan *virtua*l memori yang begitu besar . Dengan demikian pada basis data *right skewed* kinerja waktu proses algoritma CHARM tidak berbeda jauh dengan waktu proses algoritma CHARM_NEW.

Pertumbuhan C secara eksponensial seperti terlihat pada Tabel 2 menyebabkan memori yang dibutukan algoritma CHARM berkembang secara eksponensial karena CHARM menggunakan tabel *hash* untuk menyimpan seluruh C yang terbentuk sebagai dasar pemeriksaan duplikasi. Terbatasnya memori utama menyebabkan kebutuhan virtual memori Algoritma CHARM untuk minimum support yang makin kecil akan semakin besar. Penggunaan virtual memori yang makin besar menyebabkan kinerja *hash* turun (waktu proses makin lama) karena waktu akses *virtual* memori lebih besar bila dibandingkan waktu akses memori utama. Algoritma CHARM_NEW menggunakan PRESET sebagai dasar pemeriksaan duplikasi sehingga untuk nilai minimum support yang makin kecil kebutuhan memori yang dibutuhkan hanya bertambah secara linier. Penggunaan memori terbesar pada algoritma CHARM_NEW terjadi bila kombinasi semua 1-*itemset frequent* (F1) mempunyai properti 3 atau 4, sehingga tidak ada penghapusan F1 dalam proses enumerasi. PRESET yang digunakan sebanding dengan | F1 - 1 | x l. Untuk nilai minimum support yang makin kecil virtual memori Algoritma CHARM_NEW jauh lebih kecil bila dibandingkan algortima CHARM, sehingga waktu proses algoritma CHAR_NEW cenderung makin baik bila dibandingkan dengan algoritma CHARM.

**Tabel 2**. Jumlah F1, Preset, dan Closed untuk Basis Data Pumsb

| ***Basis Data pumsb*** | | | |
|---|---|---|---|
| Min_Sup | F1 | PRESET | Closed (C) |
| 80% | 25 | 20 | 33.295 |
| 70% | 34 | 29 | 241.196 |
| 60% | 39 | 34 | 1.074.627 |
| 50% | 52 | 43 | 7.121.264 |

## 5. KESIMPULAN

Hasil penelitian menunjukkan bahwa algoritma CHARM_NEW lebih efisien penggunaan memorinya bila dibandingkan dengan algoritma CHARM untuk minimum support yang makin kecil. Efisiensi memori tidak dipengaruhi oleh tipe distribusinya tetapi dipengaruhi karakteristik basis datanya, semakin kecil jumlah record basis data dengan jumlah *frequent closed itemset* makin besar maka efisiensi memorinya semakin besar.

Waktu proses algoritma CHARM_NEW untuk minimum support yang makin kecil cenderung makin baik bila dibandingkan algoritma CHARM.

## PUSTAKA

[1] C. Lucchesse, S. Orlando and R. Perego, (2005), "*Fast and Memory Efficient Mining of Frequent Closed Itemsets*," IEEE Trans. On Knowledge and Data Eng., December 2005.

[2] C. Lucchesse, S. Orlando and R. Perego, (2003), "*Kdci: a multi-strategy algorithm for mining frequent sets*," In Proc. Of the 2003 Workshop on Frequent Itemset Mining Implementations, Melbourne, Florida, USA., December 2003.

[3] C. Lucchesse, S. Orlando and R. Perego, (2002), "*Adaptive and resource aware mining of frequent sets*," In Proc. Of the IEEE Int. Conference on Data Mining, Maebashi, Japan, December 2002.

[4] M. J. Zaki and C.-J. Hsiao, (2005), "*Efficient Algorithms for Mining Closed Itemsets and Their Lattice Structure*," IEEE Trans. On Knowledge and Data Eng., vol. 17, no. 4, pp. 462-478, April 2005

[5] M. J. Zaki and K. Gouda, (2003), "*Fast Vertical Mining Using Diffsets*," Proc. Ninth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining, Aug 2003.