Perkembangan Algoritma untuk Menghitung Pola yang Sering Muncul pada Basis Data yang Besar

Rosyidah Khairun Nafisah –NIM 13506107

Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
Email: rosyidah_165@students.itb.ac.id

ABSTRAK

Perlu banyak algoritma dalam dunia perkomputasian. Begitupun pada pengoperasian dan penganalisisan data dan informasi yang selalu beredar dari waktu ke waktu.

Dalam makalah ini akan dijelaskan beberapa algoritma yang berkembang di lingkungan basis data, terutama yang berkaitan dengan *data mining*.

Pembahasan akan dimulai dari pengertian apa itu *data mining* (DM), *frequent set counting* (FSC), dan pengertian istilah lainnya yang berkaitan dengan topik ini sampai pembahasan algoritma *Direct Count and Intersect* (DCI) serta perbaikan strateginya (k-DCI).

Sumber data untuk pembuatan makalah ini adalah pustaka yang didapatkan dari jurnal internasional melalui internet.

Kata kunci: algoritma, pengoperasian dan penganalisisan data dan informasi, *data mining*, *frequent set counting*, *direct count and intersect*, pustaka, internet, jurnal internasional.

1. PENDAHULUAN

Kebutuhan manusia akan data dan informasi tidak dapat dipungkiri. Bahkan, sekarang melalui dunia teknologi, arus informasi dapat beredar dengan cepat dan mudah.

Perlu pengorganisasian dan pengontrolan dari data dan informasi yang ada agar tidak simpang siur dan sulit dipahami, apalagi jika menjadikannya tidak valid karena saat data diolah menjadi informasi, banyak persepsi yang mengganggunya.

Di sanalah peran *data mining*. *Data mining* adalah suatu teknologi untuk mengekstrak pengetahuan atau yang kita kenal sebagai informasi dari kumpulan data sehingga hasilnya bisa dipergunakan untuk pengambilan keputusan.

Dalam jurnal sumber, *data mining* diartikan sebagai sebuah ilmu mengekstrak informasi yang bermanfaat dan tidak memiliki banyak penafsiran dari sejumlah besar data yang mungkin dikumpulkan dari banyak bidang seperti ilmu sains, bisnis-industri, dan teknik atau rekayasa [1].

Dalam dunia *world wide web* atau yang kita kenal sebagai internet, *data mining* akan sangat bermanfaat karena bagaimanapun basis data bagi ruang ini adalah sangat luas.

Oleh karena manfaat *data mining* ini sangat luas, maka pengembangnya pun sangatlah banyak [2]. Sebagaimana juga dunia komputer lainnya, perkembangannya sangat cepat.

Problem *Frequent Set Counting* (FSC) sering ditemukan pada semua bagian (pola / himpunan) yang terjadi pada transaksi terakhir dari suatu basis data dimana setiap transaksi adalah sebuah variabel dari kumpulan panjang bagian (pola / himpunan) tersebut [3].

Perkembangan berikutnya dari problem ini adalah suatu algoritma yang kita sebut *Direct Count and Intersect* (DCI). Karena kebutuhan mendasar seperti kecepatan dan memori, algoritma ini pun dikembangkan lagi menjadi algortima yang lebih kompleks yang berikutnya disebut k-DCI.

2. PEMBAHASAN

2.1 Algoritma DCI dan k-DCI

Sebagai perbandingan, sebelum dibahas lebih lanjut, pada bagian ini akan disajikan dua algoritma yang telah didapatkan dari dua jurnal, yaitu jurnal induk tentang DCI dan jurnal pengembangan tentang k-DCI.

Sengaja diletakkan sejajar pada halaman berikutnya agar mudah dibaca dan dibandingkan bagian mana yang ditambahkan dari DCI sehingga menjadi k-DCI.

Bagaimanapun juga dari sini akan terlihat bahwa k-DCI benar-benar turunan dari DCI. Akan tetapi, karena ada tambahan *source*, performansinya menjadi lebih baik.

```
Input: D, s



F1 = first scan(D, s, &D2);
/* find frequent items and remove non-
frequents from D */
F2 = second scan(D2, s, &D3);
/* find frequent pairs from pruned
dataset */
k = 2;
/* until pruned dataset is bigger than
memory... */
while (Dk+1 .size() > memory
available()) do
      k + +;

Fk = horizontal iter(Dk , s, k, &Dk+1
); /*... keep the horizontal format */
end while
/* switch to vertical format */




dense = Dk+1.is dense();
/* measure dataset density */
while (Fk ≠ ø ) do
      k + +;

if (dense) then
      Fk = vertical iter dense(Dk ,
s,k);
      /* dense dataset optimization */
else
      Fk = vertical iter sparse(Dk , s,
k, &Dk+1 );
      /* sparse dataset optimization */
end if
end while
```

Algoritma 1: Direct Count and Intersect - DCI

```
Input: D, min supp

/* During first scan get optimization
figures*/
F1 = first scan(D, min supp);
/* second and following scans on a
temporary */
F2 = second scan(D0, min supp);


k = 2;


while (D0.vertical size() > memory
available()) do
      k + +;
/* count-based iteration */
Fk = DCP(D', min supp, k);

end while

k + +;
/* count-based iteration + create
vertical database VD*/
Fk = DCP(D', VD, min supp, k);

dense = VD.is dense();

while ( Fk  ≠ ø ) do
      k + +;
if (use key patterns()) then
      if (dense) then
            Fk = DCI dense keyp(VD, min
supp, k);

      else
            Fk = DCI sparse keyp(VD,
min supp, k);

      end if
else
      if (dense) then
            Fk = DCI dense(VD, min
supp, k);
      else
            Fk = DCI sparse(VD, min
supp, k);
      end if
end if
end while
```

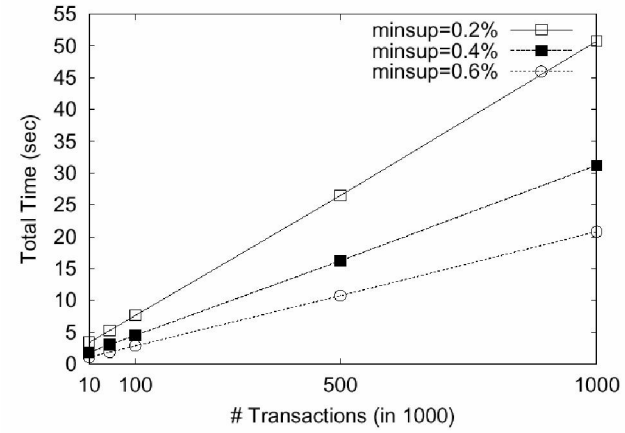
Algoritma 2: k -DCI

Pada beberapa nama fungsi yang sama terlihat k-DCI menonjolkan dihilangkannya variabel output. Selain itu, jenjang kasus kondisional bagi k-DCI di akhir bertambah dengan menggunakan kata kunci pola. Hal inilah yang mendukung performa dari k-DCI dibandingkan DCI sendiri. Jika tidak menggunakan jenjang kondisional tambahan ini, k-DCI akan sama persis dengan DCI.

Perhatikan masukan *dense* sebelum masuk kondisional terakhir. Pada DCI *dense* diinput oleh basis data D pada indeks ke k+1, sedangkan pada k-DCI *dense* diinput oleh clone basis data D yang indeksnya sengaja dinaikkan satu sebelumnya.

2.2 Beberapa Perbandingan yang Dilakukan

Pada subbab ini akan dibahas sedikit lebih jauh tentang DCI dan k-DCI. Di bawah ini adalah grafik pengaruh nilai persen minsup terhadap waktu.



Waktu yang dibutuhkan jika minimal supply naik

Ternyata, mengubah nilai minsup dapat mempengaruhi performansi tools FSC dalam hal waktu. Lebih besar pesentasi minsup, maka lebih besar juga efisiensi penggunaan waktu.

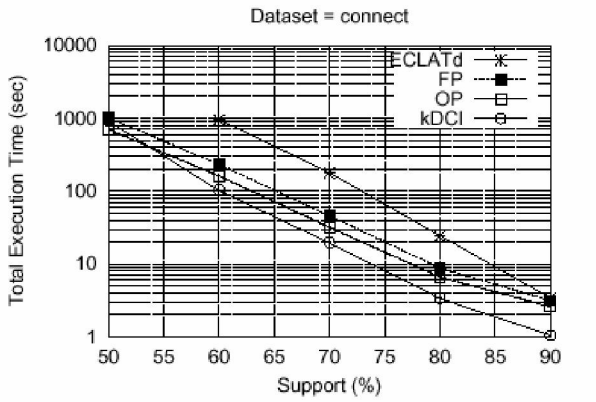
Di bawah ini adalah perbandingan DCI dengan algoritma lainnya. Pada makalah ini tidak akan dibahas lebih lanjut tentang selain k-DCI. Bagian ini hanya memperlihatkan bahwa performa k-DCI masih lebih baik daripada yang lainnya pada berbagai jenis dataset.

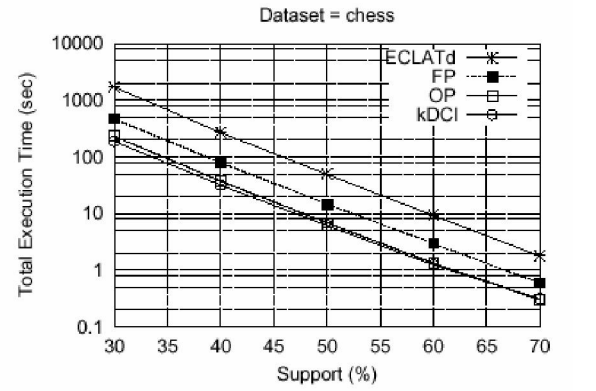
Beberapa *dataset* yang tertulis dalam pustaka yang grafiknya ada di samping adalah:

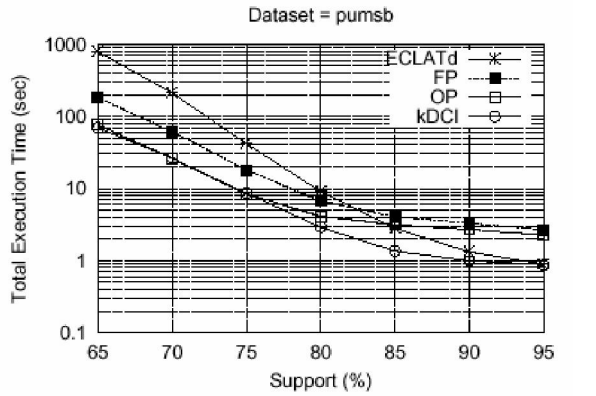
- Connect
- Chess
- Pumsb dan Pumsb_star

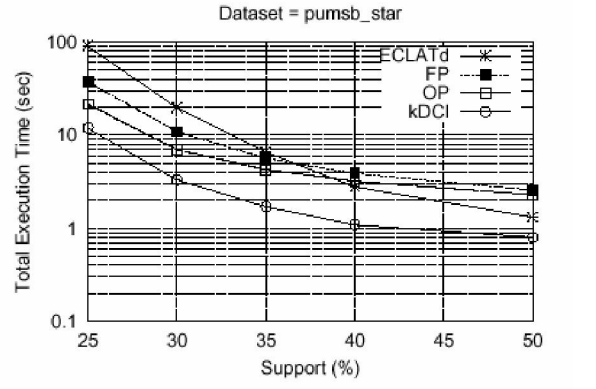
Selain itu, ada beberapa yang lainnya, seperti:

- Mushroom
- BMS_View_1
- T25I10D10K
- T30I16D400K









Pada keempat grafik di atas kDCI merupakan yang tercepat dibandingkan algoritma yang lainnya. Akan tetapi, pada empat grafik berikutnya yang tidak disajikan pada makalah ini, kDCI memiliki saingan berat. Terutama pada dua jenis terakhir, yaitu:

- T25I10D10K
- T30I16D400K

Pada dua jenis dataset ini, kDCI kalah cepat dibandingkan FP dan OP [5].

2.3 Beberapa Hal yang Berkaitan dengan k-DCI

Ada beberapa pembahasan menarik yang ada pada sumber penulisan makalah ini. Akan tetapi, penulis tidak akan menjelaskannya panjang lebar. Silakan memiliki sendiri jurnalnya langsung dari internet dalam format pdf dengan kata kunci berupa potongan dari referensi.

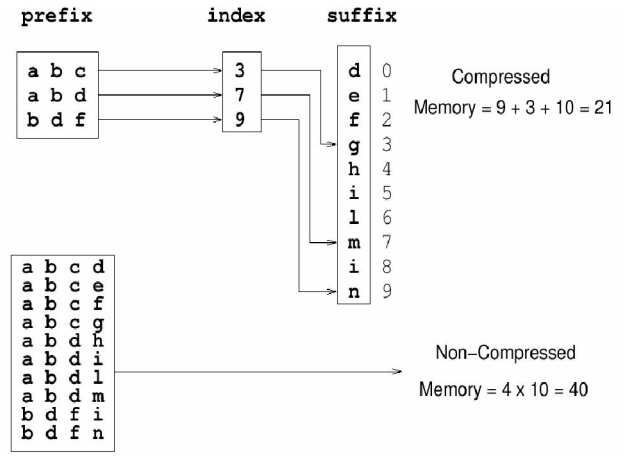
Beberapa konsep itu di antaranya adalah:

2.3.1 Dynamic data type selection

Optimasi pertama adalah dihubungkan dengan sejumlah memori yang digunakan untuk menampilkan himpunan item dan perangkatnya. Karena beberapa struktur secara terus menerus diakses selama eksekusi, yang berarti sangat menguntungkan untuk memakai memori sesedikit mungkin. Hal ini tidak hanya mereduksi kompleksitas spasial algoritma, melainkan juga menggunakan optimasi prosesor *low level* agar efektif saat *run time* [4].

2.3.2 Compressed data structures

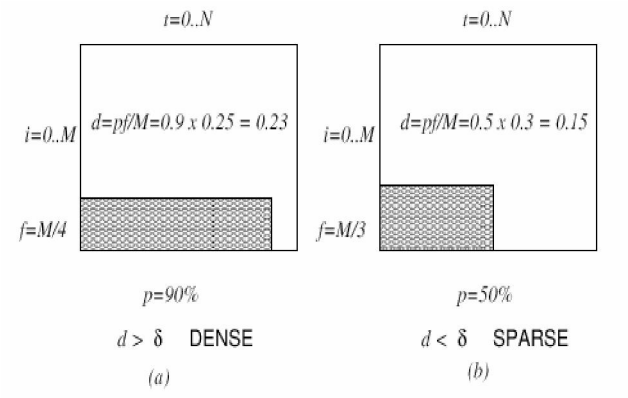
Secara tabel kira-kira dapat digambarkan seprti ini pola kompresi dengan memorinya. Contoh tabel di bawah ini dapat merepresentasikan penghematan memori sebesar 40-21 = 19. Untuk data yang sangat besar, tentu hasilnya akan sangat signifikan.



Tabel yang merepresentasikan efisiensi dengan struktur kompresi data

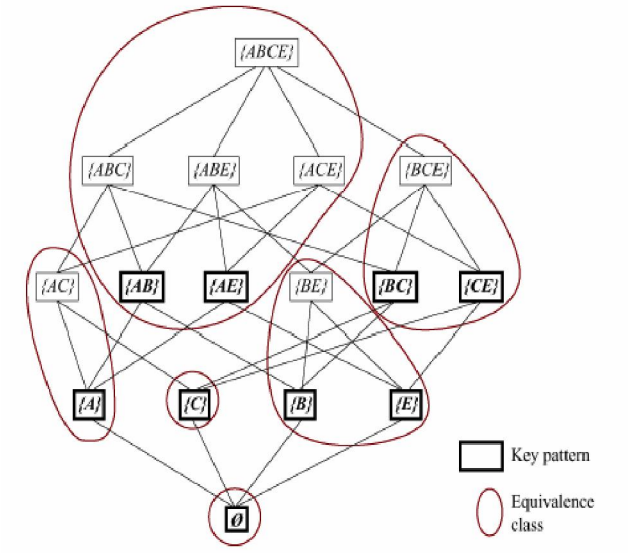
2.3.3 Heuristics

Heuristics berperan dalam membangun *density* atau *sparsity* sebuah *dataset* seperti di bawah ini:



2.3.4 Pattern Counting Inference

Ada beberapa pola yang dianggap satu kelas sehingga diklasifikasikan sama meskipun pola kunci (*key pattern*) seperti di bawah ini:



Contoh *Pattern counting inference* pada k-DCI

Pada subbab ini penulis menemukan beberapa teorema dan corrolary yang ada. Pada makalah ini akan disebutkan apa bunyi teorema yang ada, tetapi tidak akan dijelaskan lebih lanjut apa itu.

Teorema 1 Q *is a key pattern iff* supp(Q) ≠ minp Q(supp(Q \ {p})).

Teorema 2 *If* P *is not a key pattern, and* P ≤ *Q, then* Q *is a non-key pattern as well.*

Teorema 3 *If* P *is a non-key pattern and* P ≤ *Q, the following holds: f(Q) = f(Q \ (P \ P')). Where P' < P, and P with P' belong to the same equivalence class, i.e. P, P' ≤ [P].*

3. SIMPULAN

Ternyata, selama masih banyak algoritma yang belum kita pelajari. Apalagi di dunia komputer yang perubahan dan perkembangannya bisa terjadi dalam hitungan detik. Jika hanya mengandalkan kuliah, tentu akan banyak hal yang tidak kita ketahui. Oleh karena itu, mahasiswa memang dituntut lebih mandiri dalam mendalami ilmu yang ditekuninya.

Khususnya dalam makalah yang penekanannya berupa pemanfaatan strategi algoritmik pada basis data ini, simpulan yang dapat diambil adalah algoritma DCI yang sudah dianggap paling efektif pada zaman penemuannya pun sampai sekarang masih dapat diperbarui dengan menambahkan beberapa konsep dasar *heuristic*.

Masih sangat banyak yang belum dibahas dalam makalah ini. Merujuk kepada referensi utama, yaitu tesis dari seorang mahasiswa S2 di salah satu perguruan tinggi di Venezia, Italia, bernama Paolo Palmerini, yang kemudian dikembangkan lima penerusnya dari beberapa perguruan tinggi. apa yang dibahas di makalah ini adalah sebagian kecil dari sebagian kecil. Penulis sendiri masih harus banyak belajar untuk memahaminya.

Tidak menutup kemungkinan bahwa ada algoritma yang memiliki performansi lebih baik untuk problem FSC, baik itu penemuan baru maupun pengembangan dari algoritma ini. Diharapkan mahasiswa dapat lebih mengeksplorasi banyak hal di luar kuliahnya.

4. UCAPAN TERIMA KASIH

Terima kasih kepada Pak Rinaldi sebagai dosen pembimbing yang telah mengajar dan memberikan tugas pembuatan makalah ini sehingga penulis berkesempatan mengetahui hal-hal yang sebelumnya tidak penulis ketahui.

Penulis juga ingin berterima kasih kepada pihak-pihak yang telah membantu penulis menyelesaikan tugas makalah ini, terutama orang tua yang telah mengantar jemput ke warung internet malam hari saat harus browse untuk mencari topik yang tepat untuk tugas ini dan mengeksplornya lebih banyak.

Juga kepada teman-teman penulis yang begitu baik membantu baik secara langsung maupun tidak, penulis mengucapkan terima kasih banyak.

REFERENSI

[1] *On performance of data mining: from algorithms to management systems for data exploration*, 2004, halaman 1.

[2] *On performance of data mining: from algorithms to management systems for data exploration*, 2004, halaman 2.

[3] *kDCI: a Multi-Strategy Algorithm for Mining Frequent Sets*, 2006, halaman 2.

[4] *kDCI: a Multi-Strategy Algorithm for Mining Frequent Sets*, 2006, halaman 3.

[5] *kDCI: a Multi-Strategy Algorithm for Mining Frequent Sets*, 2006, halaman 4.

[6] Basis Data, Penerbit Informatika Bandung, Cetakan keempat 2002.