

# Studi Permasalahan *k-Most Promising Products* Berbasis Interval Waktu Pada Data Multidimensi Dengan Serial Waktu

Hafara Firdausi, Bagus Jati Santoso, Henning Titi Ciptaningtyas

Departemen Informatika, Fakultas Teknologi Informasi dan Komunikasi, Institut Teknologi Sepuluh Nopember (ITS)

Jl. Arief Rahman Hakim, Surabaya 60111 Indonesia

e-mail: hafarafirdausi@gmail.com<sup>1)</sup>, bagus@if.its.ac.id<sup>2)</sup>, henning@if.its.ac.id<sup>3)</sup>

**Abstrak**—Kemajuan ilmu pengetahuan dan teknologi, terutama di bidang analisis data, telah mempengaruhi cara perusahaan dalam menjalankan bisnis, yaitu dengan mengumpulkan data preferensi pelanggan dari data penjualan produk, kemudian memanfaatkannya untuk mendapatkan informasi yang dapat digunakan untuk membuat keputusan bisnis yang tepat. Saat ini, sudah ada strategi pemilihan produk dengan melakukan pencarian *k*-produk yang paling banyak diminati oleh pelanggan, yaitu *k-Most Promising Products* (k-MPP). Sayangnya, komputasi k-MPP tidak mempertimbangkan variabel waktu dalam algoritme perhitungannya dan tidak dapat digunakan untuk memproses kueri berbasis interval waktu. Artikel ini mengusulkan algoritme k-MPPTI (*k-Most Promising Products in Time Intervals*) untuk menjawab permasalahan k-MPP berbasis interval waktu pada data multidimensi dengan serial waktu. Ada tiga jenis algoritme yang dibuat dan dibandingkan, yaitu k-MPPTI (menggunakan kueri *dynamic skyline* dan *reverse skyline*), k-MPPTI NoRSL (menggunakan kueri *dynamic skyline* saja), dan k-MPPTI NoRSL-P (menggunakan teknik komputasi paralel). Efektivitas dan efisiensi algoritme diuji menggunakan data asli dan sintetis. Hasil uji coba menunjukkan bahwa algoritme k-MPPTI NoRSL memiliki performa yang lebih baik daripada algoritme k-MPPTI karena dapat memberikan hasil kueri dengan waktu eksekusi lima kali lebih cepat dan penggunaan memori satu kali lebih hemat dibandingkan dengan algoritme k-MPPTI.

**Kata kunci**—Strategi pemilihan produk, Kueri, *Dynamic skyline*, *Reverse skyline*, Interval waktu

## I. PENDAHULUAN

Pesatnya kemajuan ilmu pengetahuan dan teknologi telah mempengaruhi cara perusahaan dalam menjalankan bisnis, yaitu dengan mengumpulkan data preferensi pelanggan dari data penjualan produk, kemudian memanfaatkannya secara cerdas untuk mendapatkan informasi yang dapat digunakan untuk membuat keputusan bisnis yang tepat. Misalnya, dengan mendapatkan informasi *k*-produk yang paling diminati oleh pelanggan beserta fitur-fiturnya, perusahaan dapat menentukan harga produk baru yang akan diluncurkan atau menentukan fitur apa yang hendak diunggulkan dari produk baru yang ingin dibuat.

Saat ini, sudah ada penelitian yang mengembangkan strategi pemilihan produk dengan melakukan pencarian *k*-produk yang paling banyak diminati oleh pelanggan. Dalam [1], Islam et

al. memodelkannya sebagai kueri *k-Most Promising Products* (k-MPP) serta membuat kerangka kerja algoritme untuk memproses kueri tersebut. Komputasi k-MPP menggunakan dua tipe kueri *skyline*, yaitu *dynamic skyline* [2] dan *reverse skyline* [3]. Kueri *dynamic skyline* digunakan untuk mengambil data produk terbaik berdasarkan sudut pandang pelanggan, sedangkan kueri *reverse skyline* digunakan untuk mengambil data pelanggan potensial berdasarkan sudut pandang produk atau perusahaan [1].

Sayangnya, komputasi k-MPP tidak mempertimbangkan variabel waktu dalam algoritme perhitungannya sehingga informasi yang didapatkan kurang valid dengan kondisi yang sebenarnya. Komputasi k-MPP juga tidak dapat memproses kueri berbasis interval waktu. Sebagai contoh, pertanyaan yang mungkin diajukan adalah "*k-produk apa yang paling banyak diminati oleh pelanggan pada bulan Februari hingga September?*". Dalam hal ini, bulan Februari hingga September disebut dengan interval waktu kueri dan data yang berbasis interval waktu disebut dengan data *time series* atau serial waktu.

Sebagai ilustrasi, produk A adalah produk yang paling banyak diminati oleh pelanggan pada bulan Januari hingga Juni, namun posisinya diungguli oleh produk B yang lebih diminati pelanggan pada bulan Juli hingga September. Pada bulan Oktober, produk B tidak diproduksi lagi karena suatu alasan, sehingga produk A kembali diminati pelanggan.

Berdasarkan ilustrasi tersebut, produk yang paling unggul berdasarkan kueri k-MPP adalah produk B karena produk B pernah mengungguli produk A walaupun rentang waktu unggulnya lebih pendek daripada produk A. Hal ini terjadi karena komputasi k-MPP hanya mempertimbangkan skor kontribusi pasar yang dihitung dari banyaknya jumlah pelanggan yang lebih menyukai produk tersebut dibandingkan produk lainnya tanpa mempertimbangkan faktor durasi waktu.

Sedangkan jika berdasarkan kueri dengan interval waktu Januari hingga Juli maka produk yang paling unggul adalah produk A; jika berdasarkan kueri dengan interval waktu Juli hingga Agustus maka produk yang paling unggul adalah produk B; jika berdasarkan kueri dengan interval waktu Januari hingga Desember maka produk yang paling unggul adalah

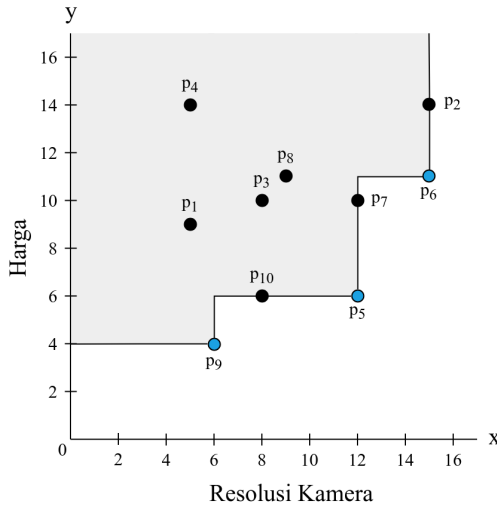
produk A karena rentang waktu unggulnya lebih lama daripada produk B.

Artikel ini membahas metode yang dapat menjawab permasalahan k-MPP berbasis interval waktu pada data multidimensi dengan serial waktu, yaitu dengan memodelkan kueri k-MPPTI (*k-Most Promising Products in Time Intervals*) dan merancang kerangka kerja algoritme yang dapat memproses kueri tersebut. Ada tiga jenis algoritme pemrosesan yang dibuat dan dibandingkan: (a) k-MPPTI, yaitu algoritme yang mengadaptasi komputasi k-MPP asli (menggunakan dua tipe kueri *skyline*, yaitu *dynamic skyline* dan *reverse skyline*); (b) k-MPPTI NoRSL, yaitu algoritme yang hanya menggunakan kueri *dynamic skyline* saja; (c) k-MPPTI NoRSL-P, yaitu algoritme k-MPPTI NoRSL yang mengimplementasikan teknik komputasi paralel supaya pemrosesan data menjadi lebih cepat. Efektivitas dan efisiensi ketiga algoritme diuji menggunakan data asli dan sintetis.

## II. TINJAUAN PUSTAKA

### A. Skyline

Operasi *skyline* digunakan untuk mencari data yang menarik dari suatu himpunan data, yaitu data yang tidak didominasi oleh data lain. *Skyline* didefinisikan sebagai titik-titik yang tidak didominasi oleh titik lain [6]; titik adalah representasi dari data dalam bidang  $d$ -dimensi. Sebuah titik  $p_1 \in P$  dikatakan mendominasi titik lain  $p_2 \in P$ , dinotasikan dengan  $p_1 \prec p_2$ , jika nilai  $p_1$  baik atau lebih baik dari  $p_2$  pada semua dimensi dan ada nilai  $p_1$  yang lebih baik dari  $p_2$  setidaknya pada satu dimensi. Secara matematis, relasi  $p_1 \prec p_2$  dapat terbentuk jika dan hanya jika: (a)  $p_1^i \leq p_2^i, \forall i \in [1, \dots, d]$  dan (b)  $p_1^i < p_2^i, \exists i \in [1, \dots, d]$ .



Gambar 1: Titik *skyline* dari data produk pada Tabel I

Sebagai ilustrasi, seseorang ingin mencari produk *smartphone* terbaik, yaitu *smartphone* yang memiliki harga termurah dan memiliki resolusi kamera terbesar. Menggunakan data produk  $P$  pada Tabel I, terdapat data produk *smartphone* yang memiliki atribut resolusi kamera ( $dim_1$ ) dan harga ( $dim_2$ ).

Setiap datanya direpresentasikan sebagai titik pada bidang dua dimensi, yakni sumbu  $x$  adalah resolusi kamera dan sumbu  $y$  adalah harga *smartphone*.

Berdasarkan Gambar 1, produk *smartphone* yang terbaik adalah  $p_5$ ,  $p_6$ , dan  $p_9$  karena tidak ada titik yang lebih baik dari titik-titik tersebut pada semua dimensi. Produk  $p_{10}$  tidak dapat menjadi *skyline* karena didominasi oleh produk  $p_5$  pada atribut resolusi kamera (sumbu  $x$ ). Begitu pula produk  $p_7$  yang didominasi  $p_5$  dan produk  $p_2$  yang didominasi  $p_6$  pada atribut harga (sumbu  $y$ ). Produk  $p_5$ ,  $p_6$ , dan  $p_9$  disebut sebagai titik *skyline* atau *skyline point*.

### B. Dominansi Dinamis

Para ahli menyebut *original skyline* sebagai *static skyline* [7] karena sifat dominansinya yang statis. Dalam pengembangannya, hasil *skyline* dapat berubah bergantung pada titik kuerinya (dominansi dinamis). Suatu titik  $ob_1 \in D$  dikatakan mendominasi titik  $ob_2 \in D$  secara dinamis berdasarkan titik kueri  $ob_3 \in D$ , dinotasikan dengan  $ob_1 \prec_{ob_3} ob_2$ , jika nilai  $ob_1$  dekat dengan  $ob_3$  pada semua dimensi dan ada nilai  $ob_1$  yang lebih dekat dengan  $ob_3$  dibandingkan nilai  $ob_2$  dengan  $ob_3$  minimal pada satu dimensi. Secara matematis, relasi  $ob_1 \prec_{ob_3} ob_2$  terbentuk jika dan hanya jika: (a)  $|ob_3^i - ob_1^i| \leq |ob_3^i - ob_2^i|, \forall i \in [1, \dots, d]$  dan (b)  $|ob_3^i - ob_1^i| < |ob_3^i - ob_2^i|, \exists i \in [1, \dots, d]$ .

Tabel I: Contoh *dataset* (a) produk  $P$  dan (b) preferensi pelanggan  $C$

(a)			(b)		
id	dim1	dim2	id	dim1	dim2
$p_1$	5	9	$c_1$	5	2
$p_2$	15	14	$c_2$	8	10
$p_3$	8	10	$c_3$	15	10
$p_4$	5	14	$c_4$	9	7
$p_5$	12	6	$c_5$	10	12
$p_6$	15	11	$c_6$	12	14
$p_7$	12	10	$c_7$	7	13
$p_8$	9	11	$c_8$	15	8
$p_9$	6	4	$c_9$	5	5
$p_{10}$	8	6	$c_{10}$	10	5

Pada Tabel I, diberikan contoh *dataset* produk dan preferensi pelanggan. Berdasarkan preferensi pelanggan  $c_1$ , produk  $p_1$  dikatakan mendominasi produk  $p_2$ , dinotasikan dengan  $p_1 \prec_{c_1} p_2$ , karena memenuhi kedua syarat dominansi dinamis yakni (a)  $|c_1^1 - p_1^1| = |5 - 5| = 0 \leq |c_1^1 - p_2^1| = |5 - 15| = 10$  dan (b)  $|c_1^2 - p_1^2| = |2 - 9| = 7 < |c_1^2 - p_2^2| = |2 - 14| = 12$ .

Sebaliknya, jika berdasarkan preferensi pelanggan  $c_6$ , maka produk  $p_2$ -lah yang mendominasi  $p_1$ , dinotasikan dengan  $p_2 \prec_{c_6} p_1$ , karena (a)  $|c_6^1 - p_1^1| = |12 - 5| = 7 \leq |c_6^1 - p_2^1| = |12 - 15| = 3$  dan (b)  $|c_6^2 - p_1^2| = |14 - 9| = 5 < |c_6^2 - p_2^2| = |14 - 14| = 0$ . Dalam hal ini, preferensi pelanggan disebut dengan titik kueri karena dapat mempengaruhi sifat dominansi antar produk.

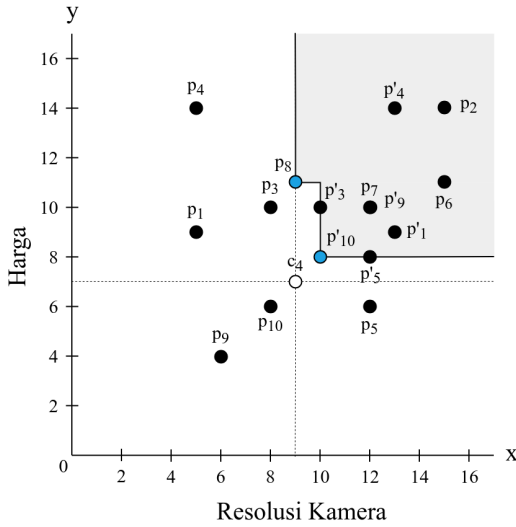
### C. Dynamic Skyline

Kueri *dynamic skyline* dalam komputasi  $k$ -MPP digunakan untuk mencari produk terbaik dari sudut pandang pelanggan

[1]. *Dynamic skyline* [2] dari seorang pelanggan  $c_1 \in C$ , dinotasikan dengan  $DSL(c_1)$ , berisi semua produk  $p_1 \in P$  yang tidak didominasi oleh produk lain  $p_2 \in P$  berdasarkan preferensi pelanggan  $c_1$ ,  $p_2 \not\prec_{c_1} p_1$ .

*Dynamic skyline* dapat dihitung menggunakan algoritme komputasi *skyline* tradisional [6] dengan cara mentransformasikan semua titik  $p \in P$  ke ruang data baru dengan menganggap titik kueri  $c$  sebagai titik asal dan jarak absolut titik  $p$  ke  $c$  digunakan sebagai fungsi pemetaan yang didefinisikan sebagai  $f^i(p^i) = |c^i - p^i|$ .

Menggunakan *dataset* pada Tabel I, *dynamic skyline* dari pelanggan  $c_4$  adalah  $DSL(c_4) = \{p_8, p_{10}\}$ , karena produk tersebut tidak didominasi oleh produk lain berdasarkan preferensi pelanggan  $c_4$ . Berbeda halnya dengan  $c_{10}$  yang memiliki hasil *dynamic skyline*  $DSL(c_{10}) = \{p_5, p_8, p_{10}\}$ .



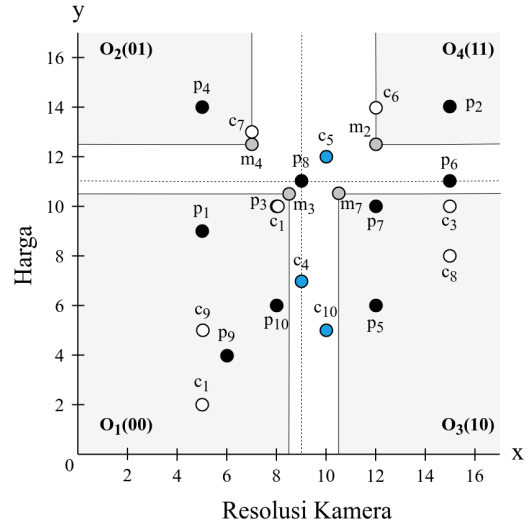
Gambar 2: Komputasi *dynamic skyline* dari pelanggan  $c_4$

#### D. Reverse Skyline

Kueri *reverse skyline* dalam komputasi *k-MPP* digunakan untuk mencari pelanggan potensial dari sudut pandang produsen [1]. *Reverse skyline* [3] dari sebuah produk  $p_1 \in P$ , dinotasikan dengan  $RSL(p_1)$ , berisi semua pelanggan  $c \in C$  yang memiliki  $p_1$  pada hasil *dynamic skyline*-nya.

Ada beberapa tahapan yang harus dilakukan dalam komputasi *reverse skyline* [1], yaitu (1) menentukan *orthant*, dinotasikan dengan  $O$ , sejumlah  $2^d$  pada data  $d$ -dimensi; (2) menghitung semua *midpoint* atau titik tengah antara produk kueri dan setiap produk  $p \in P$  menggunakan persamaan  $m_2^i = \frac{(p_1^i + p_2^i)}{2}$ ; (3) menentukan *midpoint skyline*  $MSL(o)$  (juga dikenal sebagai *mid-skyline* [8]) pada setiap *orthant*; (4) menentukan *reverse skyline* dengan mencari semua pelanggan  $c \in C$  yang tidak didominasi oleh *midpoint skyline*  $m \in M$  berdasarkan produk  $p_1$ ,  $c \not\prec_{p_1} m$ .

Sebagai contoh, berdasarkan *dataset* yang diberikan pada Tabel I, *reverse skyline* dari produk  $p_8$  adalah pelanggan  $c_4$ ,  $c_5$ , dan  $c_{10}$ , dinotasikan dengan  $RSL(p_8) = \{c_4, c_5, c_{10}\}$



Gambar 3: Komputasi *reverse skyline* dari produk  $p_8$

karena masing-masing pelanggan tersebut memiliki  $p_8$  pada hasil *dynamic skyline*-nya.

#### E. k-Most Promising Products (k-MPP)

Islam et al. memodelkan kueri *k-Most Promising Products* (k-MPP) dan merancang algoritme pemrosesan untuk memproses kueri tersebut [1]. Terdapat empat langkah pemrosesan kueri k-MPP, yaitu: (1) mencari *reverse skyline* masing-masing produk  $p \in P$ ; (2) mencari *dynamic skyline* masing-masing pelanggan  $c \in RSL(p)$ ; (3) menghitung kontribusi pasar masing-masing produk dengan cara mengakumulasi probabilitas produk dipilih oleh pelanggan, sebagaimana persamaan  $E(C, p|P) = \sum_{c \in RSL(p)} Pr(c, p|P)$ ; (4) memilih  $k$ -produk dengan kontribusi pasar terbesar.

Tabel II: Contoh data multidimensi dengan serial waktu (a) produk  $P$  dan (b) preferensi pelanggan  $C$

(a)					(b)				
ID	Timestamp		Nilai		ID	Timestamp		Nilai	
	$t_i$	$t_e$	$d_1$	$d_2$		$t_i$	$t_e$	$d_1$	$d_2$
$p_1$	2	10	6	3	$c_1$	1	8	2	8
$p_2$	6	13	4	12	$c_2$	4	14	4	10
$p_3$	9	15	6	15	$c_3$	10	15	6	11
$p_4$	4	9	9	5	$c_4$	3	8	8	12
$p_5$	5	15	12	10	$c_5$	5	15	9	10

#### F. Data Multidimensi dengan Serial Waktu

Data multidimensi dengan serial waktu adalah data *multi-attribute* yang memiliki *timestamp* dan berurutan menurut waktu, sebagaimana contoh *dataset* produk dan preferensi pelanggan yang ditunjukkan pada Tabel II. Pada tabel tersebut, *timestamp* ditulis sebagai interval waktu yang dinotasikan dengan  $[t_i : t_e]$ , dengan asumsi bahwa nilai masing-masing atribut konstan setiap waktu.

### III. METODE

Bagian ini memaparkan pendekatan yang digunakan untuk menjawab kueri k-MPP berbasis interval waktu, meliputi struktur data dan algoritma pemrosesan yang digunakan.

Tabel III: Daftar notasi

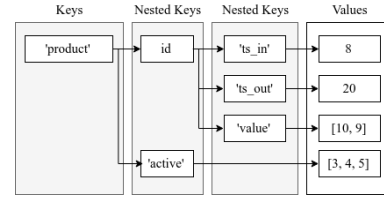
Simbol	Deskripsi
$P$	Himpunan data produk
$C$	Himpunan data preferensi pelanggan
$p$	Sebuah data produk dalam $P$
$c$	Sebuah data preferensi pelanggan dalam $C$
$D$	$P \cup C$
$ob$	Sebuah objek data pada $D$
$ob_1 \prec ob_2$	Objek data $ob_1$ mendominasi $ob_2$
$ob_1 \prec_{ob_3} ob_2$	Objek data $ob_1$ mendominasi $ob_2$ berdasarkan $ob_3$
$d$	Jumlah dimensi pada $D$
$i$	Dimensi ke-1, ..., $d$
$[t_i : t_e]$	Interval waktu
$E$	Himpunan event
$e$	Sebuah event dalam $E$ , $e \in E$
$p_i$	Data produk masuk
$p_o$	Data produk keluar
$c_i$	Data pelanggan masuk
$c_o$	Data pelanggan keluar
$PA$	Himpunan data produk yang sedang aktif
$CA$	Himpunan data pelanggan yang sedang aktif
$diff$	Selisih nilai
$O$	Orthant
$m$	Midpoint antar produk
$DSL(c)$	Dynamic skyline dari pelanggan $c$
$RSL(p)$	Reverse skyline dari produk $p$
$MSL(o)$	Midpoint skyline dari orthant $o$
$Pr_t(c, p PA)$	Probabilitas produk $p \in PA$ dibeli oleh pelanggan $c \in CA$ pada waktu $t$
$E_t(CA, p PA)$	Kontribusi pasar $p$ pada waktu $t$
$E_{[t_i:t_e]}(CA, p PA)$	Kontribusi pasar $p$ dalam interval waktu $[t_i : t_e]$
$k$	Jumlah data
$k - MPPTI$	$k$ -Most Promising Products in Time Intervals
$PB$	Pandora Box

#### A. Struktur Data

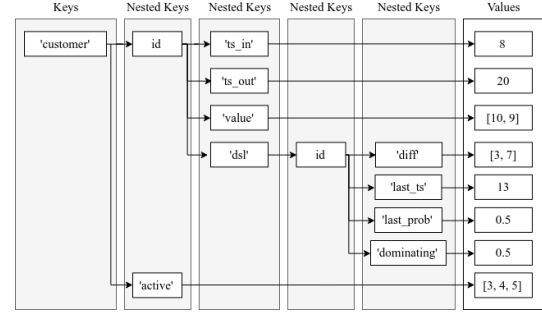
Ada tiga struktur data utama yang digunakan dalam komputasi k-MPPTI, yaitu *Data Storage*, *Event Queue*, dan *Pandora Box*.

1) **Data Storage:** Sebuah struktur data *nested dictionary* yang digunakan untuk menyimpan data produk (Gambar 4) dan pelanggan (Gambar 5). Struktur data *dictionary* lebih efisien untuk pencarian data karena menggunakan konsep *key-value pairs* dibandingkan dengan *list* atau *array* yang menggunakan indeks untuk mengakses nilai suatu data.

2) **Event Queue:** Sebuah struktur data *queue* dengan prinsip FIFO (*First In First Out*) yang berfungsi untuk menyimpan semua titik terjadinya perubahan di dalam himpunan data, yaitu jika ada data yang masuk atau keluar. Titik-titik ini disebut dengan *event*. Ada empat jenis *event* yang terjadi: (1) *Product Insertion* (data produk masuk), (2) *Product Deletion*



Gambar 4: Struktur data *dictionary* produk



Gambar 5: Struktur data *dictionary* pelanggan

(data produk keluar), (3) *Customer Insertion* (data pelanggan masuk), dan (4) *Customer Deletion* (data pelanggan keluar). Masing-masing *event* memiliki empat jenis informasi yang disimpan, yaitu *timestamp*, *role* (produk atau pelanggan), ID data, dan aksi (masuk atau keluar).

3) **Pandora Box:** Sebuah struktur data *array* dua dimensi, terdiri dari sumbu  $x$  (*time series*) dan sumbu  $y$  (produk), yang digunakan untuk menyimpan skor kontribusi pasar setiap produk pada setiap waktu. Menggunakan contoh *dataset* pada Tabel II, maka model *Pandora Box* yang terbentuk adalah seperti pada Gambar 6.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
P1	0	1	2	1	0.67	0	0	0	0	0	0	0	0	0	0
P2	0	0	0	0	0	1.33	1.33	1.33	0.5	1	1	1	1	0	0
P3	0	0	0	0	0	0	0	0	0	0.5	0.5	0.5	0.5	1	0.5
P4	0	0	0	2	1.67	1.33	1.33	1.33	0.5	0	0	0	0	0	0
P5	0	0	0	0	1.67	1.33	1.33	1.33	1	1.5	1.5	1.5	1.5	2	1.5

Gambar 6: Contoh *Pandora Box* dari dataset II

#### B. Algoritme Utama

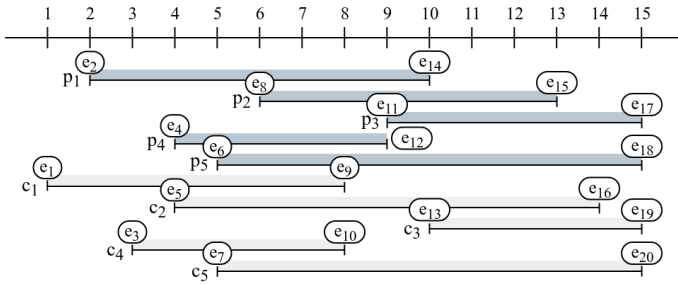
Algoritme k-MPPTI terdiri dari dua tahap pemrosesan, yaitu *data precomputing* dan *query processing*.

1) **Data Precomputing:** Tahap pertama pemrosesan yang dapat menunjang performa algoritme *query processing* supaya dapat bekerja lebih efektif dan efisien, yaitu dengan menghitung kontribusi pasar masing-masing produk berdasarkan preferensi pelanggan dan mengakumulasinya dalam *Pandora Box*. Diawali dengan mencatat semua *event* ke dalam *Event Queue*, kemudian memproses setiap *event* secara berurutan menggunakan algoritme pemrosesan tertentu berdasarkan jenis *event*-nya, dan diakhiri dengan mengeksport *Pandora Box* yang nantinya akan digunakan sebagai masukan pada tahap *query processing*.

Tabel IV: Deskripsi *key* dalam *Data Storage*

Key	Deskripsi
'product'	Menyimpan data produk
'customer'	Menyimpan data pelanggan
'id'	ID data produk atau pelanggan dijadikan sebagai <i>key</i>
'active'	Menyimpan ID data produk atau pelanggan yang sedang aktif dalam bentuk <i>array</i>
'ts_in'	Menyimpan <i>timestamp</i> atau waktu masuk
'ts_out'	Menyimpan <i>timestamp</i> atau waktu keluar
'value'	Menyimpan nilai data produk atau pelanggan pada semua dimensi dalam bentuk <i>array</i>
'dsl'	Menyimpan hasil <i>dynamic skyline</i> dalam bentuk <i>dictionary</i> dengan <i>id</i> produk sebagai <i>key</i>
'diff'	Menyimpan selisih antara nilai data produk dan pelanggan pada masing-masing dimensi
'last_ts'	Menyimpan <i>timestamp</i> terakhir saat diperbarui ke <i>Pandora Box</i>
'last_prob'	Menyimpan probabilitas terakhir saat diperbarui ke <i>Pandora Box</i>
'dominating'	Menyimpan ID produk lain yang pernah didominasi

Menggunakan *dataset* pada Tabel II, data multidimensi dengan serial waktu diilustrasikan sebagai lini masa sebagaimana pada Gambar 7. Lini masa atau alur waktu adalah suatu representasi kronologis urutan peristiwa atau kejadian (*event*) yang diwakili oleh titik-titik yang dinotasikan dengan  $e \in E$ .

Gambar 7: *Event* dalam lini masa data produk dan pelanggan

a) **Product Insertion:** Proses yang dijalankan ketika ada data produk yang masuk, dinotasikan dengan  $p_i$ . Langkah-langkah pemrosesan *product insertion* dijelaskan sebagai berikut: (1) menambahkan produk  $p_i$  ke dalam daftar produk aktif  $PA$ , (2) menghitung  $RSL(p_i)$ , (3) menghitung  $DSL(c)$  untuk setiap  $c \in RSL(p_i)$  dan menghitung probabilitas masing-masing produk  $p \in DSL(c)$ , dan (4) memperbarui *Pandora Box*. Proses *product insertion* ditunjukkan pada *pseudocode* baris 13-19 pada Algoritme 1.

b) **Product Deletion:** Proses yang dijalankan ketika ada data produk yang keluar, dinotasikan dengan  $p_o$ . Langkah-langkah pemrosesan *product deletion* dijelaskan sebagai berikut: (1) memperbarui *Pandora Box* untuk mengisi indeks  $PB$  sebelumnya jika ada yang kosong, (2) menghitung  $RSL(p_o)$ ,

**Algorithm 1** Precomputing

---

**Input** : product dataset  $P$ , customer dataset  $C$   
**Output** : pandora box  $PB$

```

1:  $EQ \leftarrow$  init event queue
2:  $D \leftarrow$  data indexing  $P, C$ 
3: for all  $d \in D$  do
4:    $EQ \leftarrow$  enqueue  $d$ 
5:  $PA \leftarrow \emptyset$  of active products
6:  $CA \leftarrow \emptyset$  of active customers
7:  $PB \leftarrow$  init pandora box
8:  $RSL \leftarrow$  init reverse skyline
9:  $DSL \leftarrow$  init dynamic skyline
10: while  $EQ$  is not empty do
11:    $e \leftarrow$  dequeue  $EQ$ 
12:   if  $e$  role is product then
13:     if  $e$  action is insertion then
14:        $PA \leftarrow$  append  $p \in e$ 
15:        $RSL(p) \leftarrow$  compute reverse skyline
16:       for all  $c \in RSL(p)$  do
17:          $DSL(c) \leftarrow$  compute dynamic skyline
18:       for all  $c \in CA$  do
19:          $PB \leftarrow$  update pandora box  $DSL(c)$ 
20:     else if  $e$  act is deletion then
21:       for all  $c \in CA$  do
22:          $PB \leftarrow$  update pandora box  $DSL(c)$ 
23:        $RSL(p) \leftarrow$  compute reverse skyline
24:       for all  $c \in RSL(p)$  do
25:          $DSL(c) \leftarrow$  find active child and
           compute new dynamic skyline
26:        $PA \leftarrow$  remove  $p \in e$ 
27:   else if  $e$  role is customer then
28:     if  $e$  act is insertion then
29:        $CA \leftarrow$  append  $c \in e$ 
30:        $DSL(c) \leftarrow$  compute initial dynamic skyline
31:        $PB \leftarrow$  update pandora box  $DSL(c)$ 
32:     else if  $e$  act is deletion then
33:        $PB \leftarrow$  update pandora box  $DSL(c)$ 
34:        $CA \leftarrow$  remove  $c \in e$ 
35: export  $PB$ 

```

---

(3) menghitung  $DSL(c)$  untuk setiap  $c \in RSL(p_o)$  yang dimaksudkan untuk mencari produk-produk yang pernah didominasi (*child*) dan menghitung probabilitas masing-masing produk  $p \in DSL(c)$ , dan (4) menghapus produk  $p_o$  dari daftar produk aktif  $PA$ . Proses *product deletion* ditunjukkan pada *pseudocode* baris 20-26 pada Algoritme 1.

c) **Customer Insertion:** Proses yang dijalankan ketika ada data pelanggan yang masuk, dinotasikan dengan  $c_i$ . Langkah-langkah pemrosesan *customer insertion* dijelaskan sebagai berikut: (1) menambahkan pelanggan  $c_i$  ke dalam daftar pelanggan aktif  $CA$ , (2) menghitung *initial*  $DSL(c_i)$  untuk mendapatkan hasil *dynamic skyline* awal dan menghitung probabilitas masing-masing produk  $p \in DSL(c)$ , dan (3) memperbarui *Pandora Box*. Proses *customer insertion* ditunjukkan pada *pseudocode* baris 28-31 pada Algoritme 1.

d) **Customer Deletion:** Proses yang dijalankan ketika ada data pelanggan yang keluar, dinotasikan dengan  $c_o$ . Langkah-langkah pemrosesan *customer deletion* dijelaskan sebagai berikut: (1) memperbarui *Pandora Box* untuk mengisi indeks  $PB$  sebelumnya jika ada yang kosong dan (2) menghapus pelanggan  $c$  dari daftar pelanggan aktif  $CA$ . Proses *customer deletion* ditunjukkan pada *pseudocode* baris 32-34 pada Algoritme 1.

e) **Dynamic Skyline**: Komputasi yang digunakan untuk mencari produk terbaik dari sudut pandang pelanggan [1]. Langkah-langkah komputasi  $DSL(c)$  secara umum adalah (1) menghitung selisih absolut dari nilai setiap dimensi antara produk dan pelanggan, dinotasikan dengan  $diff^i = |p^i - c_1^i|$ ; (2) mengecek dominansi dinamis antar produk dengan membandingkan selisih absolut-nya sebagaimana yang ditunjukkan pada Algoritme 4. Pengecekan dominansi dinamis dilakukan secara iteratif hingga dipastikan suatu  $p_1$  tidak didominasi oleh  $p_2$  lain sama sekali. Jika  $p_1$  tidak pernah didominasi, maka  $p_1$  menjadi hasil  $DSL(c)$ .

Algoritme komputasi  $DSL$  dalam k-MPPTI dibagi menjadi 3 jenis, yaitu: (1) *InitDSL* yang dijalankan jika ada data pelanggan yang masuk (*customer insertion*), ditunjukkan oleh *pseudocode* baris 1-13 pada Algoritme 2; (2) *DSL-PI* yang dijalankan jika ada data produk yang masuk (*product insertion*), ditunjukkan oleh *pseudocode* baris 14-28 pada Algoritme 2; (3) *DSL-PD* yang dijalankan jika ada data produk yang keluar (*product deletion*), ditunjukkan oleh *pseudocode* baris 29-32 pada Algoritme 2.

---

#### Algorithm 2 Dynamic Skyline Computation

---

```

Input : customer  $c$ , active products  $PA$ , product in/out  $p$ 
Output :  $DSL(c)$ 
1: procedure INITDSL( $c, PA$ ) // if customer insertion
2:    $CAND \leftarrow PA$ 
3:   sort  $CAND$ 
4:   for  $i \leftarrow 0$  to length of  $CAND$  do
5:     for  $j \leftarrow i + 1$  to length of  $CAND$  do
6:       if  $p_i \prec_c p_j$  then
7:         add  $p_j$  to the child list of  $p_i$ 
8:          $CAND \leftarrow$  remove  $p_j$ 
9:       else if  $p_j \prec_c p_i$  then
10:        add  $p_i$  to the child list of  $p_j$ 
11:         $CAND \leftarrow$  remove  $p_i$ 
12:       break
13:   return  $CAND$  as  $DSL(c)$ 
14: procedure PRODUCTIN( $c, p$ ) // if product insertion
15:    $CAND \leftarrow p$ , current  $DSL(c)$ 
16:   sort  $CAND$ 
17:    $x \leftarrow$  get index of  $p$  in  $CAND$ 
18:   for  $i \leftarrow 0$  to length of  $CAND$  do
19:     if  $i < x$  then
20:       if  $p_i \prec_c p_x$  then
21:         add  $p_x$  to the child list of  $p_i$ 
22:          $CAND \leftarrow$  remove  $p_x$ 
23:       break
24:     else if  $i > x$  then
25:       if  $p_x \prec_c p_i$  then
26:         add  $p_i$  to the child list of  $p_x$ 
27:          $CAND \leftarrow$  remove  $p_i$ 
28:   return  $CAND$  as  $DSL(c)$ 
29: procedure PRODUCTOUT( $c, p$ ) // if product deletion
30:    $ac \leftarrow$  find active childs of  $p$ 
31:    $DSL(c) \leftarrow$  call computeDSL( $c, ac$ )
32:   return  $DSL(c)$ 

```

---

f) **Reverse Skyline**: Komputasi yang digunakan untuk mencari pelanggan potensial dari sudut pandang produsen [1]. Langkah-langkah komputasi  $RSL(p)$  adalah (1) menentukan *orthant*  $O$  sejumlah  $2^d$  pada data  $d$ -dimensi yang ditunjukkan pada *pseudocode* baris 5-8 pada Algoritme 3, (2) menghitung semua *midpoint* atau titik tengah antara produk kueri dan setiap produk  $p \in P$  yang ditunjukkan pada *pseudocode*

baris 9-12 pada Algoritme 3, (3) menentukan *midpoint skyline*  $MSL(o)$  pada setiap *orthant* yang ditunjukkan pada *pseudocode* baris 13-26 pada Algoritme 3, dan (4) menentukan *reverse skyline* dengan mencari semua pelanggan  $c \in C$  yang tidak didominasi oleh *midpoint skyline*  $m \in MSL(o)$  berdasarkan produk kueri yang ditunjukkan pada *pseudocode* baris 27-34 pada Algoritme 3.

---

#### Algorithm 3 Reverse Skyline Computation

---

```

Input : product as query point  $p_q$ , active products  $PA$ , active customers  $CA$ , dimension of data  $d$ 
Output :  $RSL(p)$ 
1: procedure COMPUTERSL( $p_q$ )
2:   call DefineOrthant( $d$ )
3:   call FindMSL( $p_q, PA$ )
4:   return FindRSL( $CA$ )
5: procedure DEFINEORTHANT( $d$ )
6:   for  $i \leftarrow 0$  to  $2^d$  do
7:      $id \leftarrow$  binary of  $i$ 
8:      $o_{id} \leftarrow \emptyset$ 
9: procedure CALCMIDPOINT( $p_q, p$ )
10:  for each  $i \in d$  do
11:     $m \leftarrow \frac{(p_q^i + p^i)}{2}$ 
12:  return  $m$ 
13: procedure FINDMSL( $p_q, PA$ )
14:  for all  $p \in PA$  do
15:    if  $p \neq p_q$  then
16:       $m \leftarrow$  CalcMidpoint( $p_q, p$ )
17:       $id \leftarrow$  GetOrthantId( $p$ )
18:      if  $o_{id}$  is empty then  $o_{id} \leftarrow m$ 
19:      else
20:        for each  $mc \in MSL(o_{id})$  do
21:          if  $m \prec_{p_q} mc$  then
22:             $MSL(o_{id}) \leftarrow$  delete  $mc$ 
23:          else if  $mc \prec_{p_q} m$  then
24:            break
25:          if  $\forall mc \in MSL(o_{id}) \nprec_{p_q} m$  then
26:             $MSL(o) \leftarrow$  insert  $m$ 
27: procedure FINDRSL( $p_q, CA$ )
28:  for  $c \in CA$  do
29:     $id \leftarrow$  GetOrthantId( $c$ )
30:    if  $o_{id}$  is empty then  $RSL(p) \leftarrow$  insert  $c$ 
31:    else
32:      if  $\forall m \in MSL(o_{id}) \nprec_{p_q} c$  then
33:         $RSL(p) \leftarrow$  insert  $c$ 
34:  return  $RSL(p)$ 
35: procedure GETORTHANTID( $D$ )
36:  for each  $i \in d$  do
37:    if  $D^i \leq p_q^i$  then  $id \leftarrow$  append 0
38:    else  $id \leftarrow$  append 1
39:  return  $id$ 

```

---

g) **Metode Pengecekan Dominasi**: Metode yang digunakan untuk pengecekan dominasi adalah dengan membandingkan selisih absolut data dengan titik kueri secara iteratif sejumlah dimensi data. Sehingga, semakin banyak dimensi data maka proses pengecekan dominasi semakin lama. Metode pengecekan dominasi ini digunakan dalam setiap komputasi *dynamic skyline* dan *reverse skyline*.

h) **Perhitungan Probabilitas**: Probabilitas masing-masing produk  $p \in PA$  dipilih oleh pelanggan  $c \in CA$  dihitung menggunakan Persamaan 1.

$$Pr_t(c, p|PA) = \begin{cases} \frac{1}{|DSL(c)|} & \text{if } p \in DSL(c) \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

**Algorithm 4** Check Domination

---

**Input** : value of subject ( $ob_1$ ), value of target ( $ob_2$ ), value of query point ( $ob_3$ ), dimension of data ( $d$ )  
**Output** :  $ob_1 \prec_{ob_3} ob_2$  is true/false

```

1: procedure ISDOMINATING( $ob_1, ob_2, ob_3$ )
2:    $dominating \leftarrow 0$ 
3:    $dominated \leftarrow 0$ 
4:   for each  $i \in d$  do
5:      $diff_1^i \leftarrow |ob_1^i - ob_3^i|$ 
6:      $diff_2^i \leftarrow |ob_2^i - ob_3^i|$ 
7:     if  $diff_1^i = diff_2^i$  then
8:       continue
9:     else if  $diff_1^i < diff_2^i$  then
10:       $dominating \leftarrow dominating + 1$ 
11:     else if  $diff_1^i > diff_2^i$  then
12:       $dominated \leftarrow dominated + 1$ 
13:   if  $dominated = 0$  and  $dominating \geq 1$  then
14:     return True
15:   else
16:     return False

```

---

i) **Perhitungan Kontribusi Pasar**: Kontribusi pasar produk  $p$ , dinotasikan dengan  $E(C, p|P)$ , diperoleh dengan mengakumulasi probabilitas produk dari setiap pelanggan  $c \in RSL(p)$  sebagaimana persamaan  $E_t(CA, p|PA) = \sum_{c \in RSL(p)} Pr_t(c, p|P)$ . Skor kontribusi pasar disimpan dalam *Pandora Box* sebagaimana ditunjukkan oleh *pseudocode* baris 4-8 pada Algoritme 5. Perhitungan probabilitas dan kontribusi pasar dilakukan setiap akhir pemrosesan *event*.

**Algorithm 5** Pandora Box

---

**Input** :  $DSL(c)$ , timestamp  $ts$ , number of products  $k$ , time interval (time init  $t_i$ , time end  $t_e$ )  
**Output** : filled pandora box  $PB$ , total market contribution  $MC_p$

```

1: procedure CALCPROBABILITY( $DSL(c)$ ) // Data Precomputing
2:   for all  $p \in DSL(c)$  do // calculate probability
3:      $pr \leftarrow \frac{1}{DSL(c)}$ 
4: procedure UPDATEPB( $DSL(c)$ )
5:   for all  $p \in DSL(c)$  do
6:     if  $ts > lastts$  then
7:        $UpdateScore(p, ts, pr, lastts, lastpr)$ 
8:        $PB(p, ts) \leftarrow PB(p, ts) + pr$ 
9: procedure UPDATESCORE( $p, ts, pr, lastts, lastpr$ )
10:  for  $i \leftarrow lastts + 1$  to  $ts$  do
11:     $PB(p, i) \leftarrow PB(p, i) + lastpr$ 
12: procedure GETSCORE( $p, t_i, t_e$ ) // Query Processing
13:   $MC \leftarrow 0$ 
14:  for  $i \leftarrow t_i$  to  $t_e + 1$  do
15:     $MC \leftarrow MC + PB(p, i)$ 
16:  return  $MC$ 

```

---

2) **Query Processing**: Tahap kedua pemrosesan yang bertujuan untuk memproses kueri k-MPPTI, dinotasikan sebagai  $k - MPPTI(k, [t_i : t_e])$ , dengan memilih *subset*  $k$  produk  $P'$  dari  $P$  yang memiliki total kontribusi pasar lebih besar dibandingkan dengan *subset*  $k$  produk  $P''$  dari  $P$  yang lain dalam interval waktu pencarian. Perhitungan total kontribusi pasar dinotasikan dengan persamaan  $E_{[t_i:t_e]}(CA, p|PA) = \sum_{t=t_i}^{t_e} \sum_{c \in RSL(p)} E_t(CA, p|PA)$ . Kontribusi pasar diambil dari *Pandora Box* sebagai hasil dari *data precomputing*.

**Algorithm 6** Query Processing

---

**Input** : Pandora Box  $PB$ , number of products  $k$ , time interval (time init  $t_i$ , time end  $t_e$ )  
**Output** :  $k$  products

```

1:  $Q \leftarrow \emptyset$ 
2: for all  $p \in P$  do
3:    $MC_p \leftarrow \text{call } GetScore(p, t_i, t_e)$ 
4: sort  $MC$  in ascending order based on market contribution score
5:  $Q \leftarrow \text{get top-}k \text{ } MC$ 

```

---

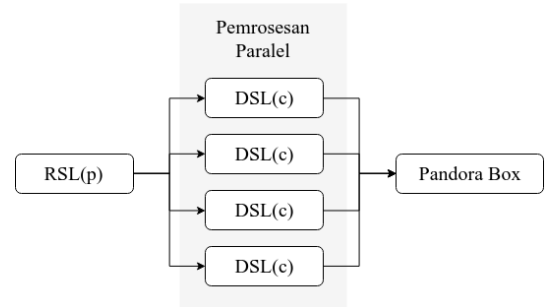
Langkah-langkah pemrosesan dalam *query processing* antara lain: (1) menghitung total kontribusi pasar setiap produk  $p \in P$  dalam interval waktu pencarian, (2) mengurutkan produk dengan total skor kontribusi pasar terbesar, dan (3) mengembalikan  $k$ -produk teratas sebagai hasil dari kueri pencarian sebagaimana yang ditunjukkan oleh *pseudocode* pada Algoritme 6.

## C. Algoritme Tandingan

Algoritme tandingan dibuat untuk membandingkan performa antar algoritme. Ada dua jenis algoritme tandingan yang dibuat, yaitu k-MPPTI NoRSL dan k-MPPTI Paralel.

1) **k-MPPTI NoRSL**: Algoritme k-MPPTI yang tidak melalui proses komputasi *reverse skyline*.

2) **k-MPPTI Paralel**: Algoritme k-MPPTI yang mengimplementasikan teknik pemrosesan paralel, yaitu suatu bentuk komputasi dua atau lebih tugas yang dilakukan secara bersamaan dan beroperasi dengan prinsip bahwa masalah besar seringkali dapat dibagi dan dipecah menjadi masalah yang lebih kecil, kemudian dipecahkan secara bersamaan (paralel) [9].



Gambar 8: Pemrosesan paralel

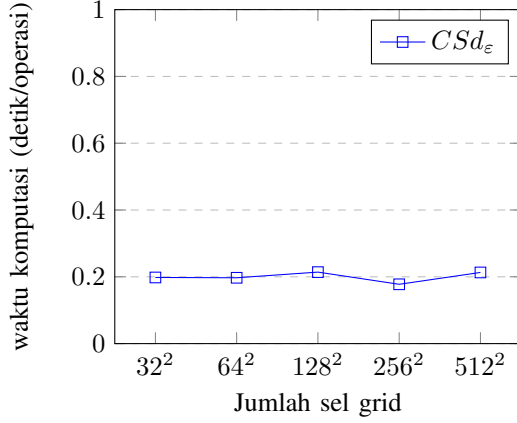
## IV. UJI COBA

Uji coba dilakukan pada jaringan jalan raya California [?] dengan *node* sebanyak 8716 dan *edge* sebanyak 9077. Algoritma diimplementasikan menggunakan bahasa Scala dengan *memory heap* sejumlah 4 GB. Pengujian dilakukan pada komputer dengan *Processor* Intel(R) Core(TM) i3-5010U CPU @ 2.10GHz x 4 dan RAM 6 GB. Pengujian dilakukan untuk mengetahui performa dengan menggunakan waktu komputasi dan untuk mengetahui penggunaan memori pada setiap eksekusi. Uji coba juga dilakukan pada tiga jenis data, yaitu data *independent*, *correlated*, dan *anticorrelated*.



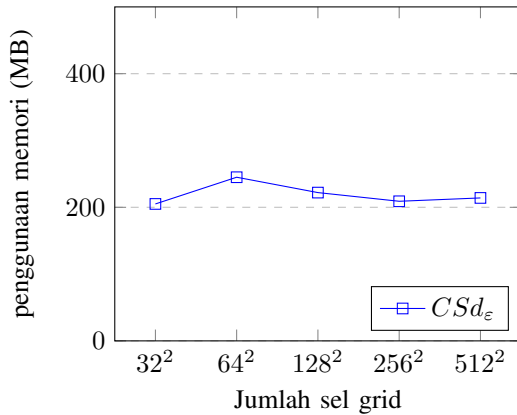
Tabel V: Variasi pengujian

Parameter	Default	Rentang
Jumlah sel grid	$256^2$	$32^2, 64^2, 128^2, 256^2, 512^2$
Jumlah objek (K)	5	0.1, 1, 5, 10, 20
Jumlah <i>instance</i> tiap objek	50	10, 50, 100, 200, 400
$d_\epsilon$ (%)	1	0.1, 0.5, 1, 2, 3
Dimensi data	2	2, 3, 4, 5, 6

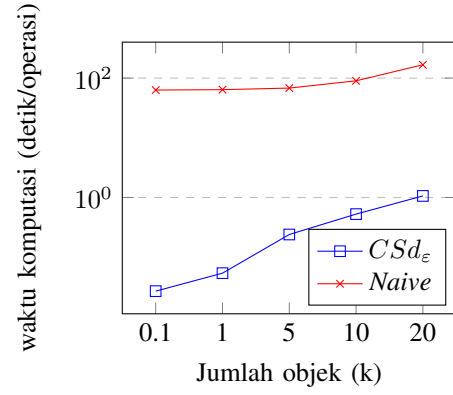


Gambar 9: Pengaruh jumlah sel terhadap waktu komputasi tiap operasi dalam satuan detik

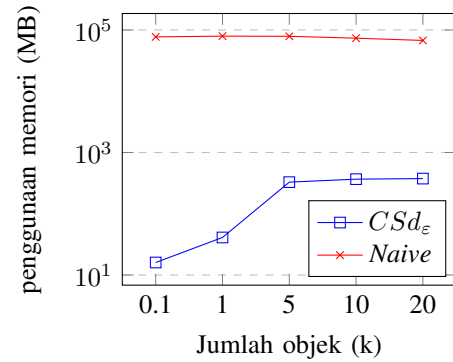
Jumlah sel tidak banyak mempengaruhi penggunaan memori dan waktu komputasi. Hal ini dikarenakan adanya *trade-off* antara proses memuat data dengan komputasi. Grid indeks yang memiliki sel sedikit menjadikan data yang dimuat lebih banyak sehingga menjadikan data yang diproses lebih banyak. Tetapi di sisi lain, sistem tidak banyak mencari data secara berulang-ulang karena setiap sel sudah mengaver area yang besar. Sedangkan grid indeks yang memiliki sel yang banyak menjadikan proses komputasi lebih efisien karena melibatkan data yang lebih sedikit. Tetapi di sisi lain, sistem harus melakukan pencarian data berulang-ulang karena sedikitnya data yang didapat pada setiap sel.



Gambar 10: Pengaruh jumlah sel grid terhadap penggunaan memori dalam satuan megabita



Gambar 11: Pengaruh jumlah objek terhadap waktu komputasi tiap operasi dalam satuan detik



Gambar 12: Pengaruh jumlah objek terhadap penggunaan memori dalam satuan megabita

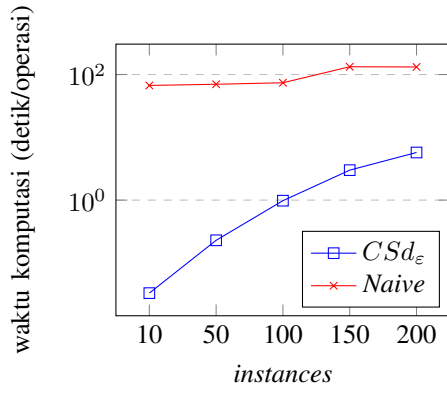
Ketika jumlah objek bertambah, waktu pemrosesan juga bertambah, hal ini dikarenakan bertambahnya objek yang terdapat pada *node*. Dengan bertambahnya objek pada *node*, algoritme perlu membandingkan dengan objek yang lebih banyak untuk mencari probabilitas masing-masing objek menjadi *SP*.

Terkait penggunaan memori, metode *naive* membutuhkan memori yang sangat banyak karena banyaknya *node* yang perlu diproses menggunakan algoritme *shortest-path*. Sedangkan metode  $CSd_\epsilon - SQ$  membutuhkan memori yang tidak banyak karena hanya menggunakan data *node* yang diperlukan saja dengan struktur grid.

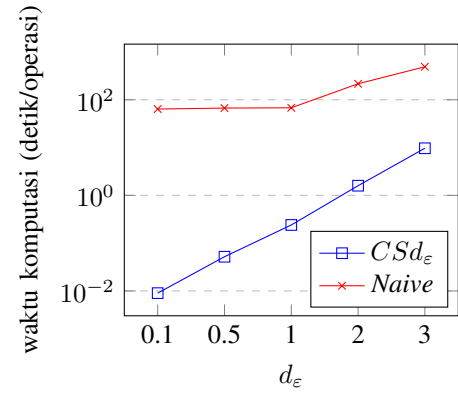
Jumlah *instance* pada objek mempengaruhi waktu komputasi dan penggunaan memori. Hal ini dikarenakan proses penghitungan probabilitas melibatkan *instances* di objek. Dari sisi memori, banyaknya *instance* membuat sistem harus mengalokasikan memori lebih untuk proses penyimpanan dan komputasi.

Pada metode *naive*, objek perubahan waktu komputasi terlihat ketika jumlah *instance* diatas 100. Hal ini dikarenakan waktu komputasi lebih banyak digunakan untuk penghitungan jarak terpendek dari setiap *node* ke objek, sehingga jumlah *instance* yang sedikit tidak berpengaruh banyak terhadap waktu komputasi.



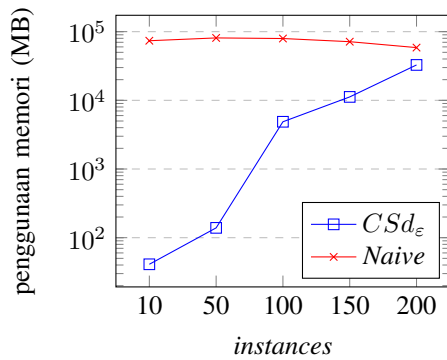


Gambar 13: Pengaruh jumlah *instance* terhadap waktu komputasi tiap operasi dalam satuan detik

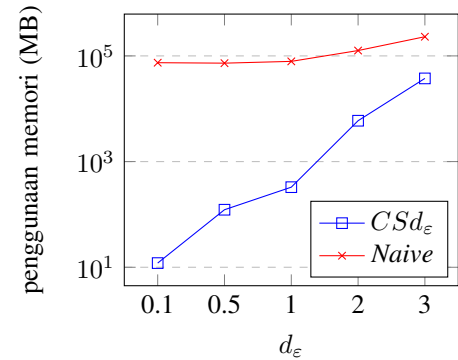


Gambar 15: Pengaruh  $d_{\epsilon}$  terhadap waktu komputasi tiap operasi dalam satuan detik

Penggunaan memori sangat tergantung dari jumlah objek yang diproses. Nilai  $d_{\epsilon}$  yang besar menjadikan objek yang diproses semakin banyak karena setiap *node* memiliki jangkauan yang lebih jauh. Banyaknya objek yang diproses menjadikan penggunaan memori semakin besar.



Gambar 14: Pengaruh jumlah *instance* terhadap penggunaan memori dalam satuan megabita



Gambar 16: Pengaruh  $d_{\epsilon}$  terhadap penggunaan memori dalam satuan megabita

Jarak  $d_{\epsilon}$  sangat mempengaruhi *performance* karena  $d_{\epsilon}$  menentukan jarak terjauh *node* yang dapat menyimpan objek baru. Dengan bertambahnya nilai  $d_{\epsilon}$ , objek dapat menjangkau lebih banyak *node*. Dengan demikian, objek yang ditampung pada *node* menjadi semakin banyak. Dengan semakin banyaknya objek, proses penghitungan probabilitas *skyline* menjadi semakin lama karena harus menghitung banyak objek. Pada  $CSd_{\epsilon}-SQ$ , semakin besar nilai  $d_{\epsilon}$ , semakin banyak grid yang diakses sehingga membutuhkan waktu yang lebih banyak. Pada metode *naive*, terdapat perubahan waktu komputasi yang signifikan ketika nilai  $d_{\epsilon}$  diatas 1.

## V. KESIMPULAN

Pada bab ini dijelaskan mengenai kesimpulan dan saran dari hasil uji coba yang telah dilakukan.

Dari proses desain hingga uji coba, dapat diambil beberapa hasil sebagai berikut:

- 1) Artikel ini mengusulkan struktur data grid indeks dan metode  $CSd_{\epsilon}$  untuk pengolahan *skyline query* pada *uncertain data streaming* oleh titik bergerak dan objek tidak bergerak. Struktur data grid indeks memecah struktur data graf tradisional menjadi sel-sel yang berisi *node*, *edge*, dan objek. Penyimpanan objek dalam bentuk *SW-Tree* pada setiap *node* membuat proses komputasi lebih cepat.
- 2) Biaya komputasi pada metode  $CSd_{\epsilon}$  jauh lebih baik dibandingkan metode *naive* dari sisi waktu komputasi dan penggunaan memori. Komputasi metode  $CSd_{\epsilon}$  lebih cepat 600 kali dibandingkan metode *naive*. Dari sisi

penggunaan memori, metode  $CSd_\epsilon$  lebih hemat 1500 kali dibandingkan metode *naive*.

Berikut beberapa saran terkait pengembangan struktur data dan algoritma lebih lanjut:

- 1) Pendefinisian jarak  $d_\epsilon$  dapat dilakukan secara dinamis. Apabila pencarian objek dengan jarak  $d_\epsilon$  tidak menemukan hasil yang diminta, jarak  $d_\epsilon$  dapat diperbesar secara dinamis hingga mendapatkan hasil yang sesuai.
- 2) Pengembangan algoritma untuk memproses objek *uncertain* yang dapat bergerak secara dinamis.
- 3) Pada algoritme ini proses pembaruan *instance* dari *uncertain* objek dilakukan dengan menghapus dan menambahkan objek baru. Hal ini tentunya tidak efisien. Diperlukan algoritme pembaruan objek agar lebih efisien dalam hal waktu komputasi dan penggunaan memori.

#### PUSTAKA

- [1] M. S. Islam and C. Liu, "Know Your Customer: Computing K-Most Promising Products," *The VLDB Journal*, pp. 545-570, 2016.
- [2] D. Papadias, Y. Tao, G. Fu and B. Seeger, "Progressive Skyline Computation in Database Systems," *ACM Transactions on Database Systems*, Vol. 30, No. 1, pp. 418-2, 2005.
- [3] E. Dellis and B. Seeger, "Efficient Computation of Reverse Skyline Queries," *VLDB Endowment*, pp. 291-302, 2007.
- [4] B. Jiang and J. Pei, "Online Interval Skyline Queries on Time Series," *IEEE International Conference on Data Engineering*, pp. 1036-1047, 2009.
- [5] M. Golfarelli and S. Rizzi, "Introduction to Data Warehousing," in *Data Warehouse Design: Modern Principles and Methodologies*, New York: McGraw-Hill, 2009, pp. 1-42.
- [6] S. Borzsonyi, D. Kossmann and K. Stocker, "The Skyline Operator," *In: ICDE*, pp. 421-430, 2001.
- [7] L. Zou, L. Chen, M. T. Zsu and D. Zhao, "Dynamic Skyline Queries in Large Graphs," *DASFAA'10 Proceedings of the 15th International Conference on Database Systems for Advanced Applications - Volume Part II*, pp. 62-78, 2010.
- [8] X. Wu, Y. Tao, R. C.-W. Wong, L. Ding and J. X. Yu, "Finding the Influence Set through Skylines," *EDBT*, pp. 1030-1041, 2009.
- [9] G.S. Almasi and A. Gottlieb, *Highly Parallel Computing*. Redwood City, CA: Benjamin-Cummings Publishers, 1989.
- [10] J. A. Blackard, D. J. Jean and C. W. Anderson, "UCI Machine Learning Repositories," 1 Agustus 1998. [Online]. Available: <https://archive.ics.uci.edu/ml/datasets/covertypes>. [Accessed 9 Juni 2018].