CONTINUOUS MULTIQUERIES K-DOMINANT SKYLINE DI JARINGAN JALAN RAYA

Syukron Rifail Muttaqi[1)] dan Bagus Jati Santoso[2)]
[1, 2)] Departemen Teknik Informatika, Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia 60111
e-mail: syukronrifai@gmail.com[1)], bagus@if.its.ac.id[2)]

ABSTRAK

Meningkatnya penggunaan perangkat mobile menjadikan data spasial patut untuk dipertimbangkan. Untuk mendapatkan hasil yang maksimal, pengguna seringkali mencari yang terbaik dari sekumpulan objek. Diantara algoritma yang dapat digunakan adalah skyline query. Algoritma tersebut mencari semua objek yang tidak didominasi oleh objek lain pada semua atributnya. Namun, data yang memiliki banyak atribut menjadikan query tersebut mengeluarkan hasil objek yang banyak sehingga kurang bermanfaat bagi pengguna. k-dominant skyline query dapat menjadi solusi untuk mengurangi objek yang menjadi hasil. Diantara hal yang menjadi tantangan adalah penggunaan skyline query dengan data spasial dan banyaknya preferensi pengguna dalam mencari objek terbaik. Penelitian ini mengusulkan IKSR: algoritma k-dominant skyline query yang bekerja pada lingkungan jaringan jalan raya serta dapat memproses banyak query yang memiliki subspace yang sama dalam sekali pemrosesan. Algoritma ini menggabungkan query yang beroperasi pada subspace dan himpunan objek yang sama dengan nilai k yang berbeda dengan cara mengkomputasi mulai dari k yang terkecil ke terbesar. Optimasi dilakukan dengan melakukan prekomputasi sebagian data untuk k yang lebih besar pada komputasi k yang terkecil sehingga komputasi voronoi cell tidak dilakukan berulang-ulang. Pengujian dilakukan dengan membandingkan dengan algoritma naïve tanpa adanya penggabungan query. Algoritma IKSR dapat mempercepat waktu komputasi dua hingga tiga kali lipat dibandingkan komputasi naïve.

***Kata kunci**: Continuous, jaringan jalan raya, k-dominant skyline, skyline.*

CONTINUOUS MULTIQUERIES K-DOMINANT SKYLINE ON ROAD NETWORK

Syukron Rifail Muttaqi[1)] and Bagus Jati Santoso[2)]
[1, 2)] Department of Informatics, Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia 60111
e-mail: syukronrifai@gmail.com[1)], bagus@if.its.ac.id[2)]

ABSTRACT

The increasing use of mobile devices makes spatial data worthy of consideration. To get maximum results, users often look for the best from a collection of objects. Among the algorithms that can be used is the skyline query. The algorithm looks for all objects that are not dominated by other objects in all of its attributes. However, data that has many attributes makes the query output a lot of objects so it is less useful for the user. k-dominant skyline queries can be a solution to reduce the output. Among the challenges is the use of skyline queries with spatial data and the many user preferences in finding the best object. This study proposes IKSR: the k-dominant skyline query algorithm that works in a road network environment and can process many queries that have the same subspace in one processing. This algorithm combines queries that operate on the same subspace and set of objects with different k values by computing from the smallest to the largest k. Optimization occurs when some data for larger k are precomputed when calculating the result for the smallest k so the Voronoi cell computing is not repeated. Testing is done by comparing with the naïve algorithm without precomputation. IKSR algorithm can speed up computing time two to three times compared to naïve algorithm.

***Keywords**: Continuous, k-dominant skyline, road network, skyline.*

I. PENDAHULUAN

Meningkatnya perangkat yang *mobile* pada beberapa tahun terakhir menjadikan penggiat bisnis dan teknologi semakin memperhitungkan data spasial untuk memaksimalkan proses bisnis. Perangkat *mobile* tersebut dapat ditemukan setiap saat dan dimanapun, seperti ponsel cerdas dan kendaraan yang sudah mengintegrasikan lokasi pada peta. Dengan demikian, perangkat-perangkat tersebut akan semakin banyak memproduksi dan menggunakan data spasial. Tidak hanya itu, secara umum data yang diproduksi dan yang digunakan oleh komputer di dunia juga semakin meningkat. Diantara tantangan yang perlu diselesaikan oleh ilmuwan adalah meningkatkan kecepatan proses ekstraksi informasi dari suatu data.

DOI: http://dx.doi.org/10.12962/j24068535.v18i2.a999

Seringkali seseorang perlu mencari objek terunggul dari sekumpulan data. Diantara algoritma yang dapat digunakan adalah *skyline query*. *Skyline query* merupakan metode pencarian yang menghasilkan objek yang tidak didominasi oleh objek lain. Dalam kata lain, objek-objek *skyline* merupakan objek yang lebih baik dari objek lain dan tidak lebih buruk dibandingkan objek yang lain pada semua atributnya. Sebagai contoh, seseorang ingin mencari hotel dengan kriteria rating yang paling tinggi dan harga yang paling rendah. Seringkali hasil dari *skyline query* lebih dari satu karena memungkinkan antar-objek tidak saling mendominasi. Sebagai contoh, hotel A memiliki rating 4,5 dengan biaya Rp400.000,00 dan hotel B memiliki rating 3 dengan biaya Rp200.000,00. Dalam hal ini, hotel A dan B tidak saling mendominasi karena tiap objek memiliki keunggulan sendiri-sendiri. Jika hotel A memiliki rating 2, maka hotel B mendominasi hotel A karena hotel B memiliki rating yang lebih tinggi dengan harga yang lebih rendah.

Dalam kehidupan sehari-hari, banyak kriteria yang dipertimbangkan untuk menentukan objek yang paling unggul. Sebagai contoh, untuk menentukan hotel yang unggul, seseorang menggunakan kriteria harga, biaya, kenyamanan, dan keamanan. Semakin banyak atribut yang terdapat pada objek maka semakin banyak kemungkinan objek yang menjadi bagian dari *skyline*. Hal tersebut disebabkan probabilitas satu objek mendominasi objek lain semakin kecil karena semakin banyaknya atribut yang menjadi kriteria.

Diantara metode untuk mengurangi jumlah objek yang menjadi *skyline* adalah *k-dominant skyline query*. *k-dominant skyline* merupakan pencarian objek *skyline* dengan kriteria suatu objek dapat menjadi *skyline* apabila terdapat *k* atribut yang sama dengan dengan *k* atribut yang sama pada semua objek lain dan terdapat minimal satu atribut yang lebih baik dari yang lain.

Di abad ke-21 ini, banyak informasi yang mengandalkan data spasial. Berbagai layanan melibatkan data spasial secara *real-time*. Sebagai contoh, layanan ojek daring dan aplikasi reservasi penginapan sangat mengandalkan informasi pengguna dan mitra berdasarkan letak geografisnya. Secara alami, permasalahan tersebut membutuhkan metode yang dapat mengkomputasi secara *real-time*. Di sisi lain, proses pengkomputasian *continuous* dapat menjadi solusi dari permasalahan tersebut yang pada umumnya banyak pengguna yang mengakses layanan umum.

Sebagaimana *skyline query* pada umumnya, *k-dominant query* dapat menggunakan jaringan jalan raya sebagai salah satu pertimbangan dalam mencari objek skyline. Diantara permasalahan yang terdapat pada jaringan jalan raya adalah pengguna atau titik *query* mungkin terdapat di berbagai tempat sehingga membutuhkan pemrosesan data yang dinamis sesuai lokasi dari pengguna. Hal tersebut menjadikan komputasi semakin berat karena terdapat banyak titik *query*. Pemrosesan secara *continuous* dapat menjadi solusi agar tidak terjadi proses komputasi yang berulang.

Secara umum, perilaku pengguna seringkali berbeda-beda. Pengguna memiliki preferensi sendiri dalam menentukan kriteria apa yang dapat dijadikan pertimbangan dalam mencari objek yang dianggap unggul. Dengan demikian, pemrosesan yang dapat mengkomputasi berbagai *query* dalam satu waktu diperlukan agar lebih efisien. Penelitian ini mengusulkan metode *multiqueries*, yaitu pemrosesan dapat menggunakan beberapa atribut sesuai preferensi tanpa harus melibatkan semua atribut dalam pemrosesannya.

Penelitian ini bertujuan untuk mengusulkan sebuah algoritma dan struktur data yang dapat memproses objek yang merupakan *k-dominant skyline* di jaringan jalan raya dengan berbagai varinasi nilai *k* secara kontinu. Pemrosesan dilakukan secara sekuensial dari nilai *k* terkecil hingga *k* terbesar dalam sekali komputasi dengan optimasi. Optimasi yang dilakukan berupa penggabungan komputasi yang berulang pada tiap penghitungan *k-skyline query*. Hal ini berbeda dengan pemrosesan *k-dominant skyline* pada umumnya yang dilakukan dengan penghitungan yang dilakukan satu per satu tiap *k*. Komputasi yang digabung tersebut hanya perlu dijalankan di awal komputasi dan digunakan pada komputasi *k* yang lain sehingga mempercepat proses komputasi secara kolektif.

Artikel ini ditulis dengan sistematika sebagai berikut: Bab 2 menjelaskan mengenai penelitian sebelumnya yang mendasari penelitian ini serta beberapa penelitian lain yang berada dalam satu topik dengan penelitian ini. Bab 3 menjelaskan mengenai algoritma dan struktur data yang diusulkan pada penelitian ini dengan didahului oleh tabel simbol yang digunakan. Bab 4 berisi skenario pengujian yang dilakukan serta evaluasi hasil pengujian. Bab 5 merupakan kesimpulan dari penelitian ini.

II. STUDI LITERATUR

Operator *skyline* kali pertama dicetuskan oleh Börzsönyi dkk. [1]. Operator *skyline* memfilter objek-objek yang menarik dari data set yang besar. Dalam definisi lain, operator *skyline* menghasilkan semua objek yang tidak didominasi oleh objek lainnya pada semua atributnya. Komputasi *skyline* juga disebut sebagai *maximum vector problem* [2]. Börzsönyi dkk. [1] mengusulkan algoritma *block-nested-loop* untuk mengatasi permasalahan ini.

TABEL I
SIMBOL DAN KETERANGANNYA.

Simbol	Keterangan
$\mathbf{O}, O$	Himpunan objek dan jumlahnya
o, o_i	Sebuah objek dan objek ke-*i*
$\mathbf{D}, D$	Himpunan dimensi dan jumlah dimensinya
$\boldsymbol{Q}$	Himpunan *query*
q, q_i	Sebuah *query* dan *query* ke-*i*
$q_i.k$	Nilai *k* dari *query* ke-*i*
$Dom(o)$	Himpunan objek yang mendominasi *o*
$Dominate(o)$	Himpunan objek yang didominasi *o*

Namun, algoritma *skyline* ini menghasilkan objek yang semakin banyak apabila jumlah dimensi yang digunakan semakin banyak. Hal tersebut menjadikan hasil dari komputasi tidak memberikan wawasan yang signifikan bagi pengguna.

Dalam ranah *skyline query* di spasial, beberapa penelitian [3] [4] [5] mengusulkan algoritma untuk menghitung *skyline* di jaringan jalan raya dengan kondisi objek yang bergerak dan statis. Fu dkk. [5] mengusulkan algoritma *index-based* untuk menyelesaikan permasalahan *continuous skyline query* di jaringan jalan raya dan membuktikan bahwa performa *index-based* lebih baik daripada *landmark-based.* Algorima *index-based* tersebut menggunakan struktur data *voronoi* untuk mencari *scope* area skyline dari setiap objek.

Chan dkk. [6] mengusulkan algoritma *k-dominant skyline* untuk mengatasi kekurangan dari *simple skyline query*, yaitu dengan mengurangi jumlah hasil dari komputasi. Selanjutnya Kontaki dkk. [7] mengusulkan algoritma *k-dominant skyline* yang bersifat *continuous* untuk memproses *data stream* dengan komputasi seminimal mungkin. Di lingkungan paralel, Zaman dkk. [8] mengusulkan algoritma *k-dominant skyline* terdistribusi yang dapat bekerja di lingkungan *multicore*. Hao dkk. [9] juga menyelesaikan hal yang sama dengan model MapReduce.

III. METODE

Pada bab ini, pertama-tama penulis memaparkan batasan masalah serta permasalahan serta landasan teori dalam definisi formal. Selanjutnya penulis memberikan analisa dari permasalahan dan solusinya. Terakhir, struktur data dan algoritma dijelaskan disertai dengan *pseudocode*-nya.

Permasalahan *k-dominant skyline* memiliki kemungkinan yang sangat luas, untuk memperjelas permasalahan yang diselesaikan serta mengurangi ambiguitas, berikut batasan masalah dari penelitian ini:

- Atribut pada objek memiliki jumlah dan nilai yang tetap atau tidak berubah-ubah.
- Setiap objek berada pada *edge* atau jalan raya.
- Tidak ada objek yang berada di tempat/koordinat yang sama persis.
- Setiap *edge* memiliki ujung *node* berupa n_s dan n_t.
- Objek dapat berpindah, masuk, atau keluar dari sistem sewaktu-waktu tanpa diketahui oleh sistem sebelumnya.
- Terdapat banyak titik *query* dan dapat berpindah sewaktu-waktu.
- Algoritma dapat melakukan komputasi pada dimensi yang sama dengan variasi nilai *k* yang berbeda.
- Jarak maksimal d_ε didefinisikan untuk membatasi jarak terjauh sebuah objek menjadi *skyline.*

A. Definisi Permasalahan

Ruang dimensi $\boldsymbol{D} = \{d_1, d_2, \ldots, d_D\}$ memiliki himpunan objek $\boldsymbol{O} = \{o_1, o_2, \ldots, o_O\}$. Objek-objek tersebut berada di jaringan jalan raya $G(V, E)$ dengan E sebagai *edge* dan V sebagai *node*. Setiap objek memiliki beberapa atribut nonspasial (contoh, kualitas dan harga) dan satu atribut spasial (yaitu, jarak).

Definisi 3.1. (Dominansi). Jika terdapat dua objek, o_i dan o_j, o_i mendominasi o_j jika semua atribut nonspasial dari o_i tidak lebih buruk dibandingkan o_j dan terdapat minimal satu atribut nonspasial pada o_i yang lebih baik dibandingkan atribut pada o_j. Secara formal, objek o_i mendominasi o_j, dinotasikan dengan $o_i \prec o_j$, jika dan hanya jika $\exists d_m \in \boldsymbol{D}, o_{i,m} \geq o_{j,m}$ dan $\exists d_n \in D, o_{i,n} > o_{j,n}$.

Definisi 3.2. (Spasial dominansi). Titik query q, objek o_i, dan objek o_j berada pada jaringan jalan raya. Objek o_i dikatakan mendominasi o_j jika i) o_i mendominasi o_j secara nonspasial dan ii) o_i mendominasi o_j secara spasial. Dalam konteks ini, o_i mendominasi o_j secara spasial jika jarak o_i titik *query* q lebih dekat dibandingkan

jarak o_j terhadap q. Secara formal, objek o_i mendominasi o_j relatif terhadap titik *query* q, dinotasikan $o_i \prec_q o_j$, jika $o_i \prec o_j$ dan $dist(o_i, q) < dist(o_j, q)$.

Definisi 3.3. (*k-dominate)*. Objek o_i *k-dominate* objek o_j apabila terdapat k atribut dengan $k < D$ di objek o_i yang tidak lebih buruk daripada k atribut di objek o_j dan terdapat minimal satu atribut di k dari objek o_i yang lebih baik dibandingkan yang di o_j. Secara formal, objek o_i *k-dominate* objek o_j, dinotasikan $o_i \prec o_j$ apabila $\exists\, \boldsymbol{D'} \subseteq \boldsymbol{D}, D' = k, \forall\, d_m \in \boldsymbol{D'}, o_{i,m} \geq o_{j,m}$ dan $\exists\, d_n \in \boldsymbol{D'}, o_{i,n} > o_{j,n}$.

Definisi 3.4. (*k-dominate* di jaringan jalan raya). Sebuah objek o_i *k-dominate* o_j, dinotasikan $o_j \prec_{kq} o_j$, di titik *query* q pada jaringan jalan raya apabila o_i mendominasi o_j dilihat dari atribut nonspasial($o_i \prec_q o_j$) dan spasial ($o_i \prec_q o_j$).

Sebagai contoh, Tabel II menampilkan lima objek dalam enam atribut/dimensi pada dataset **D**. **D** memiliki empat *simple skyline*, yaitu o_1, o_2, o_3, dan o_4. Dari keempat objek tersebut, hanya terdapat tiga *5-dominant skyline*, yaitu o_1, o_2, dan o_3. Objek o_4 di-*5-dominate* o_2.

k-dominant skyline query memungkinkan terdapatnya *cyclic dominance relationship* hingga mengakibatkan tidak adanya objek yang menjadi *k-dominant skyline*. Dalam artian, objek saling mendominasi dan membentuk hubungan dominansi lingkaran. Perhatikan Tabel III, tabel tersebut memperlihatkan *cyclic dominance relation-ship* ketika nilai k = 3. Secara spesifik, o_1 *k-dominate* o_2, o_2 *k-dominate* o_3, o_3 *k-dominate* o_4, dan o_4 *k-dominate* o_1.

Definisi 3.5. (*k-dominant skyline* di jaringan jalan raya/kSQ(k, **D'**)). Sebuah objek o_i dikatakan anggota dari *k-dominant skyline* di titik *query* q apabila objek tersebut tidak didominasi(*k-dominate*) oleh objek lain o_j dari atribut spasial maupun nonspasial. Secara formal, $\text{kSQ}(k, D') = \{O' | O' \subseteq O, \forall o' \in O', \forall o \in O, o \nprec_{kq} o', o \neq o'\}$.

Definisi 3.6. (*Continuous multiqueries k-dominant skyline* di jaringan jalan raya). Himpunan objek $\boldsymbol{O}$ yang ter-dapat pada jaringan jalan raya G(V, E) dengan himpunan dimensi **D** serta himpunan *query* **Q**. Untuk setiap *query* $\forall q_i \in \boldsymbol{Q}$ dengan dimensi $q_i.\boldsymbol{D}, q_i.\boldsymbol{D} \subseteq \boldsymbol{D}$ dan $q_i.k, q_i.k < D$, hitung *k-dominant skyline* di setiap titik p di E, $\forall p \in E$.

B. Index Based k-Dominant Skyline di Jaringan Jalan Raya(IKSR)

Ide pokok dari algoritma ini adalah memaksimalkan penggunaan voronoi diagram untuk menghitung beberapa *query* dalam satu komputasi. Algoritma ini dapat menjalankan *k-dominant skyline query* yang memiliki nilai k yang berbeda. Namun *query-query* tersebut harus berada dalam himpunan dimensi **D** yang sama dan **O** yang sama. Sebagai contoh, perhatikan Gambar 1, q_1, q_2, dan q_3 dibuat dari himpunan dimensi **D** dan himpunan *query* **Q**. Setiap *query* memiliki *subspace* $\boldsymbol{D'} \subseteq \boldsymbol{D}$ dan nilai k. Dari ketiga *query* tersebut, q_1 dan q_2 dapat dikomputasi-kan dalam sekali komputasi voronoi karena memiliki **D'** yang sama.

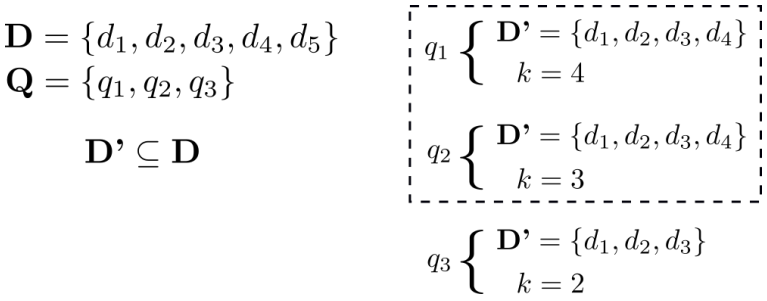
Pada subbab ini, penulis menyampaikan analisis, struktur data, dan algoritma yang digunakan untuk mengkomputasi *k-dominant skyline* di jaringan jalan raya. Sebelum membahas lebih jauh, beberapa konsep dan definisi yang digunakan pada subbab selanjutnya perlu dijelaskan.

TABEL II
CONTOH DATASET D.

Objek	d_1	d_2	d_3	d_4	d_5	d_6
o_1	5	5	5	3	3	3
o_2	3	3	3	5	5	5
o_3	4	4	4	2	4	4
o_4	2	2	2	6	2	2
o_5	3	3	3	4	4	4

TABEL III
CONTOH DATASET DENGAN *CYCLIC DOMINANCE RELATIONSHIP*.

Objek	d_1	d_2	d_3	d_4
o_1	4	4	4	4
o_2	8	3	3	5
o_3	7	8	2	2
o_4	6	7	8	1



Gambar 1. Penggabungan *query* q_1 dan q_2 karena memiliki *subspace* yang sama.

Definisi 3.7. (*Point-based Dominant Skyline Query*/PDSQ(*p, k*)). Himpunan objek **O** serta titik *query p* pada himpunan dimensi **D** terdapat pada jaringan jalan raya G(V, E). PDSQ(*p, k*) mengeluarkan semua objek, o' yang tidak di-*k-dominate* oleh objek lain *o* baik dari dimensi nonspasial maupun spasial. Secara formal, pada ruang dimensi **D**, PSQ(*p, k*) = $\{ \boldsymbol{O'} | \boldsymbol{O'} \subseteq \boldsymbol{O}, \forall o' \in \boldsymbol{O'}, \forall o \in \boldsymbol{O}, o \prec_k o', o \neq o', dist(p, o') < dist(p, o)\}$.

Definisi 3.8. (*Dominant Skyline Scope*/DSS(*o, k*)). *Dominant Skyline Scope* dari objek *o* dengan di himpunan dimensi **D** pada jaringan jalan raya G(V, E) adalah semua area pada E', $E' \subseteq E$, yang *o* menjadi anggota dari *skyline points* dari area E' tersebut dengan jarak maksimal d_ε. Secara formal, $DSS(o, k) = \{E' | E' \subseteq E, \forall\, p \in E', o \in PDSQ(p, k), dist(p, o) \leq d_\varepsilon\}$.

1) Analisis

Pertama penulis menyampaikan mengenai ciri dan sifat dari voronoi yang digunakan untuk mengkomputasi *simple skyline query* pada jaringan jalan raya. Fu dkk. [5] mengusulkan algoritma *Index Based Algorithm*(IBA) yang menggunakan *voronoi cell* untuk menyelesaikannya. *Voronoi cell* merupakan area dari suatu objek o_i yang objek tersebut merupakan objek terdekat. Objek o_i dapat dinyatakan sebagai *centroid* karena berada di tengah *voronoi cell* tersebut. Fu dkk. [5] menjadikan *voronoi cell* tersebut sebagai SS/*Skyline Scope*. Pada penelitian ini, penulis memodifikasi SS menjadi DSS/*Dominant Skyline Scope*. Perbedaannya adalah metode ini menggunakan operasi *k-dominate* untuk membandingkan dominansi antarobjek, sedangkan metode sebelumnya menggunakan operasi *simple dominate skyline*. Selanjutnya, penulis menggunakan istilah *voronoi cell* yang juga berarti DSS untuk memudahkan penjelasan.

Suatu objek o_i yang terdapat pada *E* dinyatakan sebagai *skyline* apabila o_i tidak didominasi oleh objek lain pada atribut spasial dan nonspasial. Untuk menghitung luas dari *voronoi cell* tersebut, algoritma perlu memastikan bahwa di area tersebut objek tidak didominasi objek lain. Sehingga, algoritma perlu mengetahui objek-objek lain di sekitar yang mendominasinya. Dari penjelasan tersebut, dapat disimpulkan dengan Proposisi 1.

Proposisi 1. Semakin banyak objek yang mendominasi, maka semakin banyak *voronoi cell* yang digunakan untuk proses komputasi.

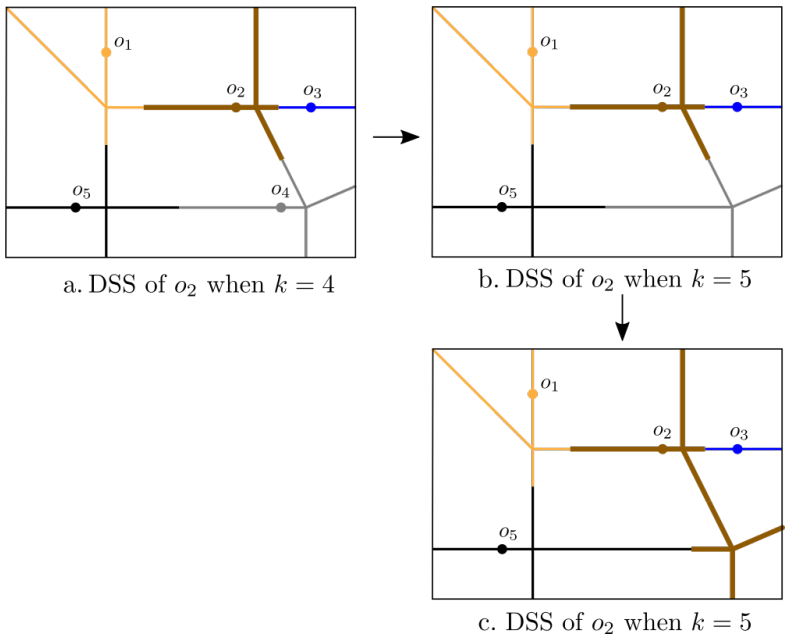
Sesuai tujuannya, *k-dominant skyline* digunakan untuk memperkecil *output* dari *skyline query* dengan mempertimbangkan *k* atribut untuk perbandingan dominansinya. Hal tersebut diungkapkan pada teorema dan *corollary* yang telah disampaikan oleh Chan dkk. [6] yang menyatakan bahwa:

Teorema 1. Jika titik p_i merupakan *k-dominant skyline*, $p_i \in DSP(k, \boldsymbol{D}, \boldsymbol{S})$, maka p_i merupakan $p_i \in DSP(k+1, \boldsymbol{D}, \boldsymbol{S})$.

Corollary 1.1. $|DSP(k, \boldsymbol{D}, \boldsymbol{S})| \leq |DSP(k+1, \boldsymbol{D}, \boldsymbol{S})|$

Dengan demikian, semakin kecil nilai *k* yang digunakan pada *k-dominant skyline query*, maka semakin sedikit jumlah *output* yang dihasilkan. Hal tersebut disebabkan pada *k* yang kecil, objek cenderung lebih mudah mendominasi jika hanya mempertimbangkan beberapa atribut dan kemungkinan terdapatnya *cyclic dominance relationship* lebih besar dibandingkan nilai *k* yang besar.

Kembali pada definisi *skyline point* pada jaringan jalan raya di [5], Fu dkk. menyatakan bahwa objek dikatakan *skyline* pada jaringan jalan raya jika tidak didominasi pada atribut nonspasial dan spasial oleh objek lain. Secara spesifik, objek *o* dikatakan *skyline point* pada titik *p* apabila tidak didominasi oleh objek lain $o', o' \in \boldsymbol{O}$ dan jaraknya tidak lebih dari jarak setiap objek yang mendominasinya secara nonspasial, $\forall\, o' \in Dom(o), dist(p, o) < dist(p, o')$. Hal tersebut senada dengan definisi *k-dominant skyline point* pada penelitian ini sebagaimana yang tertulis di Definisi 3.5. Perbedaan metode ini dengan [5] adalah proses perbandingan dominansi antarobjek pada atribut nonspasial. Untuk menyatakan dominansi, *simple skyline* menggunakan keseluruhan dimensi **D** dengan jumlah *D* sedangkan *k-dominant skyline* hanya menggunakan *k* dimensi tertentu, $k < D$. Dengan demikian, konsep pada Proposisi 1 yang bekerja untuk *simple skyline* juga dapat diaplikasikan pada kasus *k-dominant skyline*.



Gambar 2. DSS dari objek o_2 dari $k = 4$ ke $k = 5$.

Pada Corollary 1.1, semakin besar nilai *k* berarti memungkinkan semakin besar juga jumlah *skyline points*-nya. Hal tersebut disebabkan jumlah objek yang mendominasi dan didominasi semakin sedikit. Proposisi 1 juga dapat berlaku bahwa semakin sedikit objek yang mendominasi, maka semakin sedikit jumlah *voronoi cell* yang perlu dikomputasi. Dengan demikian, kesimpulan dapat diambil dengan pernyataan:

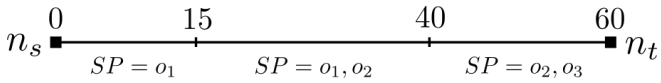
Preposisi 2. Jika semakin besar nilai *k*, maka memungkinkan adanya pengurangan *voronoi cell* pada graf voronoi.

Preposisi 3. Setiap objek/*voronoi cell* yang terdapat pada $k + 1$ graf voronoi, pasti terdapat pada *k* graf voronoi.

Terkait algoritma Parallel Dijkstra [10], algoritma ini melakukan *traverse* dari semua *centroid* -atau objek dalam konteks penelitian ini- sebagai awal *traverse* dan membuat area yang menjauh dari *centroid* untuk membuat *voronoi cell* untuk masing-masing *centroid* pada jaringan jalan raya. Berdasarkan Teorema 1, setiap objek yang merupakan *k-dominant skyline* di *k* akan menjadi *k + 1-dominant skyline*. Demikian juga yang terdapat pada komputasi graf voronoi, setiap objek menjadi *centroid* di *k + 1* juga menjadi centroid di *k* graf voronoi.

Berdasarkan uraian diatas, untuk menghitung voronoi dari *k + 1* graf voronoi, algoritma hanya perlu menghapus *voronoi cell* dari *centroid* yang tidak terdapat pada *k* graf voronoi kemudian menghitung area yang belum dikomputasi saja. Dengan demikian, komputasi graf voronoi harus dimulai dari nilai *k* yang terkecil ke nilai *k* yang terbesar atau dengan urutan *descending*. Komputasi tersebut menjadi lebih efisien karena dapat memaksimalkan beberapa komputasi Parallel Dijkstra menjadi satu kali komputasi saja. Namun, komputasi tersebut hanya berlaku jika beberapa *k-dominant skyline query* berada pada *subspace* **D** dan himpunan objek **O** yang sama sebagaimana yang terdapat pada Teorema 1.

Sebagai contoh, perhatikan Gambar 2, gambar tersebut menunjukkan kasus proses penghitungan DSS dari objek o_2 ketika $k = 3$ dari data DSS ketika $k = 4$. Sebelumnya, Gambar 2 mengasumsikan bahwa objek o_2 didominasi oleh o_1, o_3, o_4, o_5 dituliskan dengan $Dom(o_2) = \{o_1, o_3, o_4, o_5\}$ ketika nilai $k = 4$. Ketika nilai $k = 3$, objek o_2 tidak lagi di-*k-dominate* oleh o_4 sehingga $Dom(o_2) = \{o_1, o_3, o_5\}$. Gambar 2.a menampilkan DSS/*voronoi cell* dari objek o_2 yang ditandai dengan garis yang tebal. Karena objek o_4 tidak lagi mendominasi o_2, o_4 tidak lagi dikomputasi ketika $k = 5$ sebagaimana Gambar 2.b. Dengan demikian, *edge* yang sebelumnya ditempati oleh DSS dari o_4 tidak lagi digunakan. Objek o_2 dan o_5 merupakan objek yang secara langsung bersinggungan dengan DSS dari o_4 sehingga DSS dari kedua objek tersebut meluas untuk menempati DSS dari o_4.



Gambar 3. Visualisasi BTree pada *edge*.

Input: the set of objects **O** in road network $G(V, E)$, the set of queries **Q**
Output: *k-dominant skyline* BTree for each *query* in each *edge*

```
for e ∈ E do
    for q ∈ Q do
        create empty BTree for q in e;
    end
end
G = group each q ∈ Q based on it's dimentions/subspace;
for g ∈ G do
    sort q ∈ g based on q.k in ascending order;
end
for o ∈ O do
    for g ∈ G do
        for q ∈ g do
            if q.k is the smallest then
                V = compute initial voronoi graph;
            else
                V = delete voronoi cells in V;
                V = compute unused road network in V;
            end
            save voronoi V as DSS of o into BTree on each edge;
        end
    end
end
```

Gambar 4. Algoritma konstruksi untuk konstruksi awal *k-dominant skyline*.

Terdapat satu ciri yang dimiliki oleh *k-dominant skyline*, yaitu kemungkinan adanya *cyclic dominance relationship* [6]. Hal ini tidak terjadi di kasus yang terdapat pada penelitian ini, yaitu *k-dominant skyline* di jaringan jalan raya. Sebagaimana yang tertulis pada Definisi 3.5, terdapat dua properti yang menjadikan sebuah objek bagian dari *k-dominant skyline*, yaitu objek tidak didominasi oleh objek lain dalam hal atribut spasial dan nonspasial. Jika metode ini hanya mempertimbangkan atribut nonspasial, maka ia memungkinkan terdapatnya *cyclic*. Namun, metode ini juga mempertimbangkan atribut spasial, yaitu jarak, sebagai faktor dominansi. Jarak setiap objek $\forall\, o \in \boldsymbol{O}$ pada suatu titik p, $dist(o,p)$ tidak memungkinkan terjadinya *cyclic*. Sebagai contoh, tidak mungkin terdapat tiga objek yang jaraknya saling mendominasi seperti $d_a < d_b < d_c < d_a$. Hal tersebut tidak akan terjadi. Dengan demikian, jika tidak dibatasi oleh jarak maksimal d_{max}, maka setiap titik p di jaringan jalan raya, $\forall\, p \in E$, terdapat minimal satu objek yang menjadi *k-dominant skyline*.

2) Struktur Data dan Algoritma

Metode yang diusulkan pada penelitian ini bersifat *continuous*, yaitu algoritma ini akan mengkomputasi hasil *k-dominant skyline* di semua area yang terdapat pada jaringan jalan raya dan tidak tergantung dari jumlah dan lokasi *query* yang diminta oleh pengguna. Metode ini melakukan *indexing* pada setiap *edge*/jalan di jaringan jalan raya sehingga ketika pengguna mencari informasi mengenai *skyline* di suatu titik $p \in E$, algoritma hanya perlu melakukan *lookup* pada hasil *indexing* tersebut.

Hasil dari komputasi *k-dominant skyline* ini disimpan dalam struktur data BTree sebagaimana yang terdapat pada [5]. Setiap *edge* di jalan raya memiliki struktur BTree untuk setiap *query* yang berupa <*key*, SP>. BTree melakukan *indexing* pada *key*-nya dan membentuk struktur *tree* dengan kompleksitas O(log n) untuk proses pencarian/*lookup*. *Key* pada struktur tersebut merupakan jarak maksimal SP(himpunan objek yang menjadi *skyline points*) menjadi *skyline*.

Sebagai contoh visualisasi representasi *skyline points* pada BTree, perhatikan Gambar 3. Objek o_1 menjadi *skyline* di titik 0 atau n_s hingga titik 40, objek o_2 dari jarak 15 hingga jarak 60 atau n_t, dan objek o_3 menjadi *skyline* dari jarak 40 hingga akhir atau n_t. Struktur tersebut direpresentasikan dalam BTree sebagai berikut dengan setiap poin sebagai *node* yang terdapat pada BTree:

- $< 0, \{o_1\} >$
- $< 15, \{o_1, o_2\} >$
- $< 0, \{o_1, o_3\} >$

Dengan demikian, setiap *node* dari BTree hanya perlu menyimpan awal jarak dari SP dan SP tersebut tetap valid hingga ada *node* setelahnya. Jika *node* tersebut *node* terakhir dari BTree, maka SP tersebut valid hingga n_s/ujung akhir dari *edge*. BTree tersebut juga menyimpan metadata berupa *subspace* **D** dan nilai *k* dari *query* yang digunakan untuk menghasilkannya.

```
   Input: object o in inserted to G(V, E)
   Output: updated k-dominant skyline in G(V, E)
1  for g ∈ G do
2      for q ∈ g do
3          if q.k is the smallest then
4              V = compute initial voronoi graph;
5          else
6              V = delete voronoi cells in V;
7              V = compute unused road network in V;
8          end
9          save voronoi V as DSS of o into BTree on each edge;
10     end
11     for each object m ∈ Dom(o) do
12         Q' = ∅;
13         for q ∈ G do
14             if o (q.k)-dominate m then
15                 append q into Q';
16             end
17         end
18         remove DSS of m from related BTree for each q ∈ G;
19         for q ∈ Q' do
20             if q.k is the smallest then
21                 V = compute initial voronoi graph;
22             else
23                 V = delete voronoi cells in V;
24                 V = compute unused road network in V;
25             end
26             save voronoi V as DSS of m into BTree on each edge;
27         end
28     end
29 end
```

Gambar 5. Algortma *insertion*: masuknya objek *o* pada jaringan jalan raya.

Gambar 4 menunjukkan algoritma pembuatan *k-dominant skyline* yang menghasilkan BTree dari setiap *query* pada setiap *edge*. Pertama, setiap *query* dikelompokkan berdasarkan dimensinya q.**D**. Hal tersebut diperlukan karena komputasi voronoi satu *subspace* dapat dilakukan dalam sekali komputasi. Selanjutnya, setiap *query* di *group* perlu disortir dari nilai *k* yang terkecil hingga terbesar karena komputasi pembentukan voronoi berjalan dari nilai *k* terkecil.

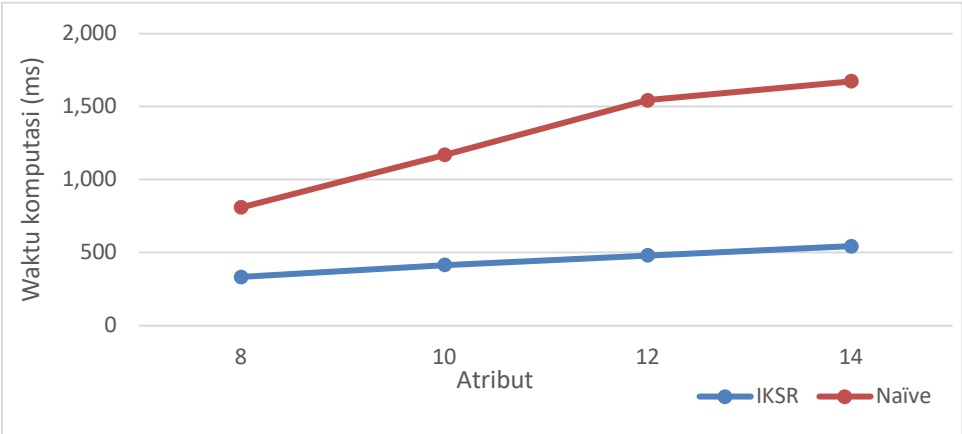
Selanjutnya untuk setiap objek yang terdapat pada jaringan jalan raya dikomputasikan *query* pada setiap *group*. Pada baris ke-13, jika *q* merupakan *query* pertama, maka ia menghitung *voronoi cell* dari objek *o* tersebut secara keseluruhan. Kemudian, pada baris ke-15, jika *query* bukan pertama, maka ia perlu menghapus *voronoi cell* yang tidak digunakan pada komputasi dengan nilai *k*. Hal ini perlu mengingat bahwa pada (k + 1)-*dominant skyline* terdapat kemungkinan bahwa objek tidak lagi mendominasi sebagaimana pada *k-dominant skyline*. Terakhir, simpan *voronoi cell* dari objek *o* sebagai DSS dengan metadata dari *query q* pada setiap BTree di *edge* yang sesuai.

Selanjutnya penulis memaparkan algoritma untuk proses *insertion*, yaitu masuknya objek dalam sistem dan menghitung *k-dominant skyline* di setiap *edge* untuk setiap *query*.

Gambar 5 menampilkan struktur data untuk proses *insertion*, yaitu objek *o* masuk pada sistem dan menghitung *k-dominant skyline* di setiap *edge* yang terlibat. *Edge* yang terlibat pada $G(V,E)$ adalah *edge* yang berjarak kurang dari sama dengan d_ε. Proses *insertion* hampir sama dengan proses konstruksi seperti Gambar 4, perbedaannya pada proses *insertion* terdapat langkah tambahan untuk menghitung ulang DSS dari objek-objek yang didominasi oleh objek baru. Secara spesifik, baris 2-10 menghitung DSS dari objek baru *o* pada semua *query* dan menyimpan dalam bentuk BTree. Kemudian di baris 11, algoritma mencari setiap objek yang didominasi oleh objek baru *o*. Pada baris 13-17, algoritma mencari setiap *query* yang objek *o* mendominasi objek *m* dan memasukkannya pada **Q'**. Algoritma kemudian menghapus semua DSS yang berkaitan dengan objek *m* karena DSS tersebut akan dihitung ulang. Selanjutnya, pada baris 19-26. Algoritma melakukan proses yang sama seperti proses awal *insertion*.

Gambar 6. Grafik pengaruh jumlah objek terhadap waktu komputasi dalam *miliseconds*.

Gambar 7. Grafik pengaruh jumlah objek terhadap waktu komputasi dalam *miliseconds*.

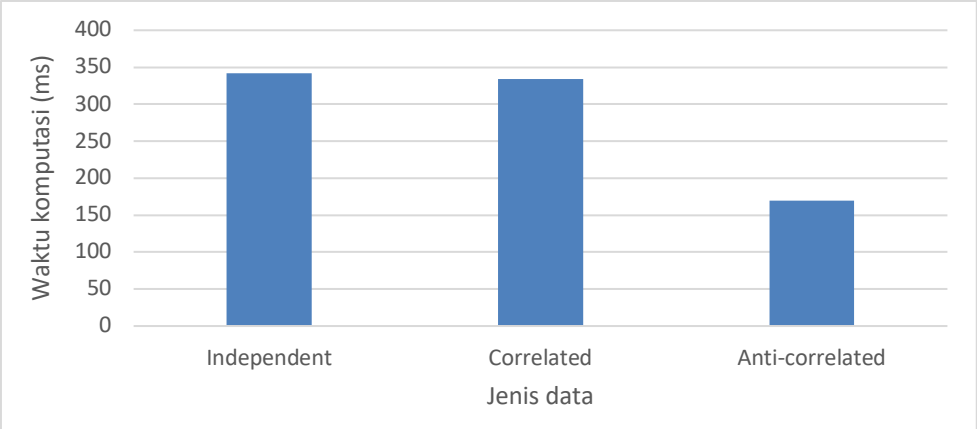


Gambar 8. Grafik pengaruh jumlah atribut terhadap waktu komputasi dalam *miliseconds*.

Proses *deletion* hampir sama seperti proses *insertion*, kecuali di awal. Proses *insertion* yang menghitung objek baru (baris 2-10 Gambar 5) diganti dengan penghapusan objek di setiap Btree sebagaimana pada baris 18. Proses komputasi ulang DSS objek yang didominasi tetap dihitung karena objek-objek tersebut tidak lagi didominasi sehingga DSS mereka bertambah besar.

IV. Hasil dan Pembahasan

Struktur data dan algoritma diimplementasikan menggunakan bahasa pemrograman Rust. Pengujian dijalankan pada laptop HP Envy 13 dengan RAM 16 GB, berprosesor Intel Core i7-1050U, dan sistem operasi Pop!_OS 19.04. Pengujian menggunakan data peta California [11]. Peta tersebut memiliki node sejumlah dinormalisasi sehingga jarak terjauh objek di sumbu x adalah 100 satuan dan begitu juga dengan sumbu y. Objek yang digunakan dibuat secara sintetis yang terbagi menjadi tiga, yaitu *independent*, *correlated*, *anti-correlated* sebagaimana yang terdapat pada [1]. Algoritma IKSR dibandingkan dengan algoritma *k-dominant skyline* tanpa ada



Gambar 9. Grafik pengaruh jenis data terhadap waktu komputasi dalam *miliseconds*.

penggabungan *query* dengan *subspace* yang sama, penulis menyebutnya dengan algoritma *naïve*. Algoritma *naïve* melakukan komputasi tanpa ada hubungan antara setiap *query*.

Parameter pengujian pada penelitian ini tertulis pada Tabel IV. Secara *default*, pengujian menggunakan nilai yang ditebalkan. Pengujian hanya mengambil data waktu eksekusi data dalam proses *insertion* karena fokus dari penelitian ini adalah pemrosesan dengan *continuous*. Waktu eksekusi diukur dengan memasukkan 300 objek kemudian mencari rata-rata dari masing-masing proses *insertion* tersebut. Proses *insertion* mengkomputasi *k-dominant skyline* terbaru dengan nilai *k* terendah hingga tertinggi dalam sekali komputasi. Nilai *k* terendah adalah $\left\lfloor \frac{D}{2} \right\rfloor$+1 dan *k* tertinggi adalah $D - 1$.

Gambar 6 merupakan grafik hubungan antara jumlah objek dengan waktu komputasi. Grafik IKSR maupun *naïve* mengalami kenaikan waktu komputasi karena semakin banyak objek yang didominasi, dengan demikian objek yang didominasi tersebut perlu dikomputasi ulang karena adanya perubahan *skyline scope*-nya. IKSR memiliki performa yang lebih baik dibandingkan *naïve*. Hal tersebut disebabkan IKSR menghitung *voronoi cell* untuk setiap objek hanya pada komputasi *k* terkecil. Komputasi *k* yang lebih besar hanya perlu menghapus objek yang tidak diperlukan dan komputasi area yang kosong. Di sisi lain, algoritma *naïve* menghitung *voronoi cell* secara keseluruhan pada semua komputasi *k*.

Hubungan antara jarak maksimal/d_ε dengan waktu komputasi digambarkan pada Gambar 7. Secara umum, waktu komputasi algoritma *naïve* dua kali lebih lama dibandingkan IKSR. Perubahan jarak maksimal pada pengujian mengakibatkan komputasi semakin lama dalam dua cara. Pertama, karena jarak semakin jauh berarti algoritma perlu melakukan *traverse* yang lebih lama. Kedua, jarak yang lebih jauh berarti objek dapat menjangkau objek lain yang lebih jauh untuk didominasi.

Pengaruh jumlah atribut terhadap waktu komputasi terdapat pada Gambar 8. Penambahan dua atribut tidak menaikkan waktu komputasi secara signifikan, hal tersebut disebabkan komputasi untuk perbandingan dominansi antarobjek hanya dilakukan sekali di sebelum komputasi *voronoi cell*. *Cost* untuk penghitungan dominanti lebih kecil dibandingkan *cost* untuk melakukan *traverse* pada jaringan jalan raya.

Gambar 9 menampilkan pengaruh jenis data terhadap waktu komputasi. Waktu komputasi data *independent* dan *correlated* tidak memiliki perbedaan yang signifikan karena objek *independent* cenderung terdapat dominansi pada nilai *k* yang bervariasi dan objek *correlated* juga terdapat dominansi dengan nilai *k* yang mendekati *D*. Data *anti-correlated* memiliki perbedaan yang signifikan karena sedikit terdapat relasi dominasi sehingga tidak banyak *voronoi cell* yang perlu dikomputasi. Dengan demikian, objek pada data *anti-correlated* tidak memiliki *voronoi cell* yang lebih besar dibandingkan data pada *independent* dan *correlated*.

V. KESIMPULAN

Penelitian ini mengusulkan sebuah algoritma IKSR yang *continuous* untuk komputasi *k-dominant skyline* di jaringan jalan raya. Algoritma ini menggabungkan *query* yang peroperasi pada subspace **D** dan himpunan objek **O** yang sama dengan nilai *k* yang berbeda dengan cara mengkomputasi mulai dari *k* yang terkecil ke terbesar. Komputasi *k* yang terkecil mengkomputasi sebagian yang diperlukan pada *k* yang lebih besar sehingga komputasi *voronoi cell* tidak dilakukan berulang-ulang. Komputasi ini dapat meningkatkan waktu komputasi dua hingga tiga kali lipat dibandingkan algoritma *naïve* yang tidak menggunakan penggabungan *query*.

DAFTAR PUSTAKA

[1] S. Borzsony, D. Kossmann, dan K. Stocker, "The Skyline operator," dalam *Proc. International Conference on Data Engineering*, 2001.

[2] H. T. Kung, F . Luccio, dan F. Preparata, "On finding the maxima of a set of vectors," *Journal of the ACM,* vol. 22, no. 4, hal. 469–76, 1975.

[3] B. Xu, J. Feng, dan J. Lu, "Continuous Skyline Queries for Moving Objects in Road Network based on MSO," dalam *Proc. International Conference on Ubiquitous Information Management and Communication*, 2018.

[4] Y.-K. Huang, C.-H. Chang, dan C. Lee, "Continuous distance-based skyline queries in road networks," *Information Systems,* vol. 37, 2012.

[5] X. Fu, X. Miao, J. Xu, dan Y. Gao, "Continuous range-based skyline queries in road networks," *World Wide Web,* vol. 20, no. 6, hal. 1443-1467, 2017.

[6] C.-Y. Chan, H. V. Jagadish, K.-L. Tan, A. K. H. Tung, dan Z. Zhang, "Finding k-dominant skylines in high dimensional space," dalam *Proc. ACM SIGMOD international conference on Management of data*, 2006.

[7] M. Kontaki, A. N. Papadopoulos, dan Y. Manolopoulos, "Continuous k-dominant skyline computation on multidimensional data streams," dalam *Proc. ACM symposium on Applied computing*, 2008.

[8] A. Zaman, M. M. Islam, M. A. Siddique, dan Y. Morimoto, "Distributed k-dominant skyline queries," dalam *Proc. International Conference on Computer and Information Technology*, 2012.

[9] H. Tian, M. A. Siddique, dan Y. Morimoto, "An Efficient Processing of k-Dominant Skyline Query in MapReduce," dalam *Proc. International Workshop on Bringing the Value of "Big Data" to Users* , 2014.

[10] M. Erwig, "The graph Voronoi diagram with applications," *Networks,* vol. 36, hal. 156-163, 2000.

[11] F. Li, D. Cheng, M. Hadjieleftheriou, G. Kollios, dan S.-H. Teng, "On Trip Planning Queries in Spatial Databases," dalam *Proc. International Symposium on Spatial and Temporal Databases*, 2005.