TESIS – KI142502

MENJAWAB *WHY-NOT QUESTION* PADA *K-MOST PROMISING PRODUCT* ($K - MPP$) DENGAN PENDEKATAN *DATA REFINEMENT*

Vynska Amalia Permadi
NRP. 5116201002

DOSEN PEMBIMBING

Tohari Ahmad, S.Kom., M.IT., Ph.D
NIP: 197505252003121002

Bagus Jati Santoso, S.Kom., Ph.D
NIP: 198611252018031001

PROGRAM MAGISTER
RUMPUN MATA KULIAH KOMPUTASI BERBASIS JARINGAN
DEPARTEMEN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN KOMUNIKASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2018

[Halaman ini sengaja dikosongkan]

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Komputer (M.Kom.)
di
Institut Teknologi Sepuluh Nopember Surabaya

oleh:
Vynska Amalia Permadi
NRP. 5116201002

Dengan judul :
Menjawab *Why-Not Question* pada *K-Most Promising Product* ($K - MPP$)
Dengan Pendekatan *Data Refinement*
Tanggal Ujian : 20 Juli 2018
Periode Wisuda : September 2018

Disetujui oleh:

Tohari Ahmad, S.Kom., M.IT., Ph.D.
NIP. 197505252003121002 (Pembimbing 1)

Bagus Jati Santoso, S.Kom., Ph.D.
NIP. 198611252018031001 (Pembimbing 2)

Royyana Muslim I, S.Kom., M.Kom., Ph.D.
NIP. 197708242006041001 (Penguji 1)

Dr.Eng. Radityo Anggoro, S.Kom, M.Sc.
NIP. 198410162008121002 (Penguji 2)

Waskitho Wibisono. S.Kom, M.Eng, Ph.D.
NIP. 197410222000031001 (Penguji 3)

Dekan Fakultas Teknologi Informasi,

Dr. Agus Zainal Arifin, S.Kom., M.Kom
NIP. 197208091995121001

[Halaman ini sengaja dikosongkan]

Menjawab *Why-Not Question* Pada *K-Most Promising Product* *($K-MPP$)* Dengan Pendekatan *Data Refinement*

Nama Mahasiswa : Vynska Amalia Permadi

NRP : 5116201002

Pembimbing I : Tohari Ahmad, S.Kom., M.IT., Ph.D
Pembinbing II : Bagus Jati Santoso, S.Kom., Ph.D

ABSTRAK

K-Most Promising Product ($K-MPP$) adalah strategi *product selection* yang digunakan pada proses pencarian K-produk yang paling banyak diminati oleh *customer*. Dasar komputasi yang digunakan untuk melakukan perhitungan $K-MPP$ adalah dua tipe *skyline query*, yaitu: *dynamic skyline* dan *reverse skyline*. Penentuan $K-MPP$ dilakukan pada layer aplikasi, yang merupakan layer paling atas pada model OSI. Salah satu fungsi layer aplikasi adalah untuk menyediakan layanan terbaik sesuai dengan keinginan *user*.

Dalam implementasi $K-MPP$, akan muncul suatu keadaan dimana produsen mungkin kurang puas dengan *query result* yang dihasilkan pada proses pencarian di sistem *database* (*why-not question*), sehingga mereka juga ingin mengetahui mengapa sistem *database* memberikan hasil pencarian *query* yang tidak sesuai dengan harapannya. Sebagai contoh, produsen ingin mengetahui mengapa suatu *data point* tertentu yang tidak diharapkan (*unexpected data*) muncul di *query result*, dan mengapa produk yang diharapkan (*expected data*) tidak muncul sebagai *query result*. Permasalahan yang muncul selanjutnya adalah, sistem *database* tradisional tidak dapat memberikan fasilitas analisis data dan solusi untuk menjawab *why-not question* yang diajukan oleh *user*.

Untuk meningkatkan *usability* pada sistem *database*, penelitian ini dilakukan dengan tujuan menjawab *why-not* $K-MPP$ dan memberikan solusi berupa *data refinement* dengan mempertimbangkan *user feedback* sehingga *user* dapat mengetahui mengapa himpunan hasil yang muncul tidak sesuai dengan harapan, dan dapat membantu *user* untuk memahami serta mengubah *query* agar menghasilkan *query result* sesuai keinginan *user* namun dengan *cost* perubahan seminimal mungkin. Berdasarkan proses evaluasi yang telah dilakukan terhadap tiga jenis tipe data yang berbeda, yaitu: independen, *anti-correlated*, dan *forest cover type* rata-rata waktu yang dibutuhkan untuk mencari variasi *data refinement* cenderung konstan dan akan mengalami peningkatan pada kardinalitas data dan selisih *K* yang tinggi. Rentang waktu yang dibutuhkan berada pada nilai 1.13 s hingga 3.48 s, dimana besarnya nilai rata-rata waktu dipengaruhi oleh jumlah data, dimensi data, dan selisih *K* antara $K-MPP$ dan *why-not point*.

Kata kunci: *Dynamic Skyline, Data Refinement , $K-MPP$, Reverse Skyline, Sistem Database, Why-not K-MPP*

[Halaman ini sengaja dikosongkan]

Data Refinement Approach For Answering Why-Not Question in K-Most Promising Product (K-MPP)

Student Name : Vynska Amalia Permadi

NRP : 5116201002

Supervisor I : Tohari Ahmad, S.Kom., M.IT., Ph.D

Supervisor II : Bagus Jati Santoso, S.Kom., Ph.D

ABSTRACT

K-Most Promising (K-MPP) product is an optional product selection strategy that used in the process of determining the most demanded products by consumers. The basic computations used to perform K-MPP calculations are two types of skyline queries: dynamic skyline and reverse skyline. K-MPP selection performed on the application layer, which is the last layer of the OSI model. One of the application layer functions is to provide services as the user's preferences.

In the K-MPP implementation, there will be a situation in which the Manufacturer may be less satisfied with the query results generated by the database search process (why-not question), so they also want to know why the database gives query results that do not match their expectations. For example, manufacturers want to know why a particular data point (unexpected data) appears in the query result set, and why the expected product (expected data) does not appear as a query result. The next problem is that traditional database systems will not able to provide data analysis and solution to answer why-not questions preferred by users.

To improve the usability of the database system, this study was conducted with the aim of answering why-not K-MPP and provide data refinement solutions by considering user feedback, so users can also find out why the result set does not meet their expectations, and help users to understand the result by performing analysis information and data refinement suggestion. Based on the evaluation process that has been done on three different types of data, namely: independent, anti-correlated, and forest cover type, the average time needed to find variations in data refinement tends to be constant and will increase in a large number of data cardinality and ΔK. The average evaluation time needed is vary from 1.13 s to 3.48 s, and it is influenced by the amount of data, data dimensions, and K difference between K-MPP and why-not points.

Keywords*: Dynamic Skyline, Data Refinement, K-MPP, Reverse Skyline, Database System, Why-not K-MPP, Real Time Query*

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Puji syukur kepada Allah SWT yang telah melimpahkan rahmat, taufik dan hidayah-Nya sehingga Tesis yang berjudul "Menjawab *Why-Not Question* pada *K-Most Promising Product* (K-MPP) Dengan Pendekatan *Data Refinement*" dapat diselesaikan dengan lancar dan tepat waktu.

Melalui kesempatan ini, penulis ingin menyampaikan rasa hormat dan terima kasih penulis yang sebesar-besarnya kepada semua pihak yang telah memberikan bantuan baik lahir maupun batin selama penulisan Tesis ini. Oleh karena itu, pada kesempatan ini penulis ingin menyampaikan rasa hormat dan terima kasih penulis kepada:

1. Orang tua saya, Bapak Ir. Devy Permadi, MM dan Ibu Lissa Andriyati. Adik saya Muh. Daffa' Lazuardi Permadi. Serta keluarga besar lainnya yang tiada henti-hentinya memberikan dukungan, doa, semangat, dan kasih sayang demi terselesaikannya Tesis ini.
2. Bapak Tohari Ahmad, S.Kom., MIT., Ph.D dan Bapak Bagus Jati Santoso, S.Kom., Ph.D selaku dosen pembimbing yang telah berkenan membagi ilmu dan memberikan saran serta pengarahan selama proses penyelesaian Tesis ini.
3. Bapak Waskitho Wibisono, S.Kom., M.Eng., Ph.D. selaku dosen penguji dan Ketua Program Pasca Sarjana Teknik Informatika ITS.
4. Bapak Royyana Muslim I, S.Kom, M.Kom, Ph.D dan Bapak Dr.Eng. Radityo Anggoro, S.Kom., M.Sc. selaku dosen penguji yang telah memberikan saran serta pengarahan dan koreksi dalam pengerjaan Tesis ini.
5. Seluruh dosen Departemen Teknik Informatika Fakultas Teknologi Informasi dan Komunikasi atas kesediaannya membagi ilmu yang bermanfaat kepada penulis.
6. Seluruh civitas akademika Departemen Teknik Informatika Fakultas Teknologi Informasi dan Komunikasi yang telah banyak memberi bantuan dan dukungan selama penulis menempuh studi dan selama penyelesaian Tesis ini.
7. Teman-teman Pascasarjana Departemen Teknik Informatika angkatan 2016, terima kasih atas segala bantuan, motivasi, dan kebersamaannya selama menempuh studi di Departemen Teknik Informatika ITS.

8. Semua pihak yang telah membantu dan berbagi ilmu dalam penyelesaian Tesis, yang tidak dapat penulis sebutkan satu per satu.

Penulis menyadari bahwa dalam penyusunan buku Tesis ini masih banyak kekurangan baik dalam hal penulisan maupun isinya. Oleh karena itu, saran dan kritik yang membangun dari para pembaca senantiasa penulis harapkan guna pengembangan diri. Semoga Tesis ini dapat memberikan manfaat bagi semua pihak, amin.

Surabaya, Juli 2018

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

BAB 1
PENDAHULUAN

Pada bab ini akan dijelaskan mengenai beberapa hal dasar dalam pembuatan penelitian yang meliputi latar belakang, perumusan masalah, tujuan, manfaat, kontribusi penelitian, dan batasan masalah.

1.1. Latar Belakang

Perkembangan teknologi informasi dan komunikasi pada beberapa dekade terakhir telah mengakibatkan munculnya berbagai macam proses digitalisasi informasi. Dengan melakukan pengolahan data pada komputer menggunakan algoritma tertentu, maka berbagai pengetahuan dan informasi baru yang sebelumnya mungkin belum disadari dapat diberikan. Sebagai contoh, data penjualan perusahaan pada sistem *database* dapat dievaluasi dengan melakukan proses komputasi maupun analisis secara otomatis menggunakan algoritma tertentu. Dengan begitu, perusahaan dapat memiliki wawasan mengenai strategi marketing yang dapat digunakan untuk memasarkan produknya.

Berdasarkan contoh implementasi teknologi informasi diatas, salah satu bagian yang berperan penting dan berhubungan dengan pengolahan data sebagai informasi dan evaluasi adalah sistem *database*. Dengan terus berkembangnya arsitektur dan teknik evaluasi pada sistem *database*, eksekusi dan pemrosesan *query* saat ini dapat diberikan secara *real time* tanpa terkendala oleh besarnya data yang perlu dievaluasi. Secara umum, *query* adalah pertanyaan atau kebutuhan informasi yang diinginkan oleh *user*.

Penelitian mengenai sistem *database* sebagian besar membahas mengenai efisiensi sistem dalam eksekusi *query* maupun pembagian penggunaan *resource* agar dapat memberikan kemampuan yang terbaik. Namun, tidak semua *end user* memahami keilmuan tentang sistem *database*. Maka dari itu akan terdapat permasalahan pada saat dibutuhkan evaluasi *query result* pada sistem *database*, dimana hanya *user* tertentu saja yang dapat melakukan evaluasi *query result* apabila dibutuhkan (Jagadish, et al., 2007).

Untuk meningkatkan *usability* sistem *database*, perlu diperhatikan ekspektasi *user* akan sistem *database* yang interaktif dan informatif. Apabila *query result* tidak

sesuai dengan yang diinginkan, diharapkan *user* dapat dengan mudah melakukan evaluasi tanpa diperlukan pengetahuan yang mendalam mengenai sistem *database.* Selain itu, diharapkan juga sistem *database* dapat memberikan penjelasan singkat dan informatif agar *user* dapat memahami dan mengevaluasi permasalahan yang menyebabkan munculnya *query result* tersebut. Dengan diberikannya penjelasan singkat dan informatif mengenai *query result* maka diharapkan *user* dapat lebih mudah mengevaluasi dan menentukan perbaikan *query*-nya agar hasil pencarian yang diberikan oleh sistem *database* sesuai dengan yang diharapkan. Hal ini juga tentunya akan memberikan alternatif efisiensi pencarian bagi *user* serta penghematan *resoure database* karena *user* tidak perlu melakukan pencarian berkali-kali hingga hasil yang diinginkan muncul sebagai *query result*.

Salah satu contoh implementasi rekayasa sistem *database* yang cukup menarik terdapat pada penelitian (Islam & Liu, 2016) yaitu perumusan *framework K-Most Promising Product* ($K-MPP$) sebagai strategi *product selection* yang digunakan pada proses pencarian produk yang paling banyak diminati oleh *customer*. Dasar komputasi yang digunakan untuk melakukan perhitungan $K-MPP$ adalah dua tipe *skyline query*, yaitu: *dynamic skyline* dan *reverse skyline*. *Skyline operator* pertama kali diusulkan oleh Borzony et al (2001). Dengan dilakukannya pemrosesan *query* menggunakan *skyline operator*, maka akan diperoleh data yang nilainya unik atau tidak terdominasi oleh data lain berdasarkan tiga jenis fungsi yang dapat dipilih, yaitu MIN atau minimal, MAX atau maksimal, dan DIFF atau *different*.

Dalam implementasi $K-MPP$, akan muncul suatu keadaan dimana produsen mungkin kurang puas dengan *query result* yang dihasilkan pada proses pencarian, sehingga mereka juga ingin mengetahui mengapa sistem *database* memberikan hasil pencarian *query* yang tidak sesuai dengan harapannya. Sebagai contoh, produsen ingin mengetahui mengapa suatu *data point* tertentu yang tidak terduga (*unexpected product*) muncul di *query result set* yang selanjutnya disebut *why point*, dan mengapa produk yang diharapkan (*expected product*) tidak muncul sebagai *query result* yang selanjutnya disebut *why-not point*. Permasalahan lainnya adalah, sistem *database* tradisional yang tidak dapat memberikan fasilitas analisis data dan solusi untuk menjawab *why-not question* yang diajukan oleh *user* seperti yang diilustrasikan pada permasalahan diatas.

Berdasarkan identifikasi permasalahan diatas, Liu et al (2016) telah melakukan identifikasi *causality* yang merupakan penyebab munculnya *expected data* dan *unexpected data* pada *query result* serta *responsibility* yang merupakan nilai pengaruh dari kemunculan *expected data* atau *unexpected data* pada *probabilistic reverse skyline query*. Sebagai pengembangan lebih lanjut, pada penelitian tersebut juga diimplementasikan proses identifikasi *causality* dan *responsibility* pada *reverse skyline query*. Evaluasi dilakukan dengan melakukan pengujian efektifitas dan efisiensi dari proses identifikasi *causality* dan *responsibility*. Namun, untuk menyelesaikan *why-not question*, dibutuhkan langkah lebih lanjut seperti modifikasi data atau *query* agar *expected data* dapat muncul pada *query result*, seperti yang dibahas pada penelitian (Islam, 2013), (Islam, et al., 2013), dan (Liu, et al., 2016).

Islam (2013) mengusulkan *framework* yang diberi nama FlexIQ untuk menjawab *why-not* dan *why question* pada SPJ *query result*. Dengan melibatkan *user input* sebagai *feedback*, ditentukan *query* baru yang dapat memasukkan *why-not point* dan mengeliminasi *why-point* yang tidak diharapkan muncul pada *query result*. Sebagai evaluasi efisiensi diusulkan dua metode penentuan *query* yang berbeda, yaitu: *baseline algorithm* (TBA) dan *trade-off algorithm* (TOA).

Islam et al (2013) membahas mengenai solusi yang dapat digunakan untuk menjawab *why-not question* pada *reverse skyline query*. Solusi yang diusulkan terdiri dari tiga bagian, yaitu: identifikasi *data point* yang menyebabkan *expected data* tidak muncul sebagai *reverse query result*, modifikasi *data point* atau modifikasi *query* agar *expected data* muncul sebagai *reverse query result*, serta modifikasi *data point* dan modifikasi *query*. Pada penelitian ini, evaluasi dilakukan pada *dataset* yang hanya memiliki dua nilai atribut, atau 2D-*dimensional data*, dan tujuan dari evaluasi yang dilakukan adalah membandingkan efektifitas dan performa dari tiga usulan modifikasi terhadap kardinalitas data.

Liu et al (2016) membahas mengenai solusi yang dapat digunakan untuk menjawab *why-not* dan *why question* pada *reverse top-k query*. Solusi yang diusulkan untuk menjawab *why-not question* hampir mirip dengan penelitian (Islam, et al., 2013), yaitu dengan melakukan tiga modifikasi yang berbeda: modifikasi *query*, modifikasi *data point weight* dan nilai K, serta modifikasi ketiganya. Pada penelitian ini digunakan lima *setting* dimensi yang berbeda sebagai salah satu evaluasi algoritma yang diusulkan

terhadap dimensi data, selain itu dilakukan juga evaluasi terhadap efektifitas dan performa metode usulan terhadap kardinalitas data.

Pada penelitian ini akan dilakukan analisis penyebab *why-not point* tidak muncul sebagai $K - MPP$ dan menjawab *why-not question* yang muncul pada *query result* $K - MPP$ dengan melakukan modifikasi nilai data pada *query* atau *data refinement*. Diharapkan juga modifikasi data memiliki *cost* perubahan yang seminimal mungkin.

1.2. Perumusan Masalah

Rumusan masalah yang diangkat dalam penelitian ini adalah sebagai berikut.

1. Bagaimana cara menganalisis dan menentukan penyebab *why-not point* tidak muncul sebagai *query result* pada $K - MPP$?
2. Bagaimana merancang metode untuk menentukan *data refinement* pada komputasi *reverse skyline query* untuk menjawab *why-not* $K - MPP$ dengan *cost* modifikasi yang minimal?
3. Bagaimana pengaruh metode *data refinement* yang diusulkan terhadap pemberian solusi permasalahan *why-not* $K - MPP$?

1.3. Tujuan

Tujuan yang ingin dicapai dalam penelitian ini adalah menghasilkan model pendekatan baru untuk menjawab *why-not question* yang muncul dalam *query result* $K - MPP$ dengan memberikan alternatif *data refinement* yang memiliki *cost* seminimal mungkin sehingga dapat membantu *user* dalam menentukan *query*-nya yang baru.

1.4. Manfaat

Manfaat dari penelitian ini adalah memberikan solusi informatif kepada perusahaan dan pengguna sistem *database* yang menggunakan $K - MPP$ sebagai evaluasi penjualan maupun penentuan strategi marketing dengan memaksimalkan hasil pencarian *query* menggunakan *skyline query*.

1.5. Kontribusi Penelitian

Berdasarkan penjelasan singkat mengenai $K - MPP$ dan permasalahan *why-not question* pada sistem *database*, kontribusi pada penelitian ini adalah :

1. Menjawab *why-not question* yang muncul pada $K - MPP$ yang belum dibahas pada penelitian (Islam & Liu, 2016).
2. Mengusulkan pendekatan *data refinement* untuk menjawab *Why-not question* pada *query result* $K - MPP$
3. Mengusulkan cara pemilihan *data refinement* terbaik yang memiliki *cost* perubahan seminimal mungkin agar dapat menghasilkan *query result* sesuai dengan keinginan *user*.

Dengan kontribusi yang diusulkan pada penelitian ini, diharapkan pendekatan *data refinement* dapat memberikan alternatif solusi untuk menjawab *why-not question* agar *why-not point* atau *expected data* dapat menjadi anggota $K - MPP$.

1.6. Batasan Masalah

Batasan masalah pada penelitian ini adalah:

1. Evaluasi dilakukan pada *dataset* sintetis independen (IND) dan *anti-correlated* (ANT) serta pada *dataset real* yaitu *Forest Cover Type* yang hanya berisi karakter numerik berupa bilangan bulat positif .
2. *Skyline query result* akan berisi data yang nilainya unik atau tidak terdominasi berdasarkan fungsi MIN.
3. Penelitian ini berfokus untuk menjawab satu data *why-not question* pada *query result* $K - MPP$ dengan pendekatan *data refinement* yang memiliki *cost* perubahan data minimal agar *why-not point* dapat muncul sebagai *query result* pada $K - MPP$.
4. Hasil metode yang diusulkan berupa saran *data refinement* pada salah satu dimensi data dengan nilai *cost* minimal dan merupakan salah satu anggota *query result* pada $K - MPP$.
5. Metode yang diusulkan diimplementasikan pada Matlab versi 2017a.

[Halaman ini sengaja dikosongkan]

BAB 2
KAJIAN PUSTAKA DAN LANDASAN TEORI

Pada bab ini akan dijelaskan tentang dasar teori dan pustaka yang terkait dengan penelitian. Dasar teori yang terkait adalah mengenai *skyline query processing*, dan pustaka yang terkait adalah seputar K-*Most Promising Product* ($K - MPP$), dan *why-not question* pada *query result* sistem *database*.

2.1. Landasan Teori

2.1.1 *Skyline Query*

Penelitian Borzsony et al (2001) pertama kali membahas penggunaan s*kyline operator* pada *database* dengan mengusulkan penambahan *SKYLINE OF* pada *query* SQL yang fungsinya mirip dengan *ORDER BY*. Dengan menggunakan SKYLINE OF, *query result* akan berisi data yang nilainya unik atau nilai yang tidak terdominasi dengan nilai lain berdasarkan fungsi *MIN*, *MAX*, atau *DIFF* yang dipilih pada tiap dimensi data. Contoh penggunaan *SKYLINE OF* dapat dilihat pada Gambar 2.1, dimana d_1 dan d_m adalah dimensi data atau atribut yang akan ditentukan *skyline*-nya, dan *MIN*, *MAX*, atau *DIFF* adalah fungsi yang diinginkan (*MIN* akan menampilkan nilai minimal, *MAX* menampilkan nilai maksimal, dan *DIFF* menampilkan nilai yang berbeda dari seluruh data pada dimensi yang dievaluasi).

```
SELECT ...  FROM ...  WHERE ...
GROUP BY ...  HAVING ...
SKYLINE OF [DISTINCT] d1 [MIN | MAX | DIFF],
           ...,        dm [MIN | MAX | DIFF]
ORDER BY ...
```

Gambar 2.1. *Syntax* SQL Menggunakan *SKYLINE OF*

(Borzsony, et al., 2001)

```
SELECT *
FROM Hotels
WHERE city = 'Nassau'
SKYLINE OF price MIN, distance MIN;
```

Gambar 2.2. Implementasi *Skyline Query* Menggunakan *SKYLINE OF* (Borzsony, et al., 2001)

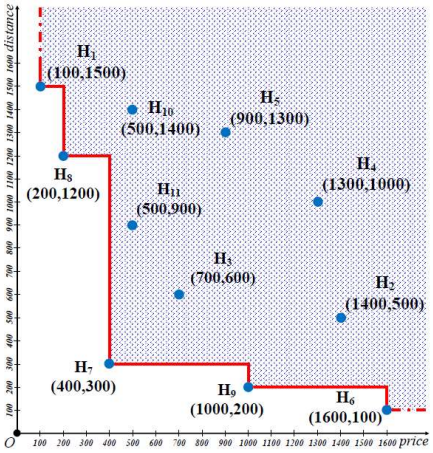
Selain menggunakan *SKYLINE OF* seperti yang ditunjukkan pada Gambar 2.2, implementasi *skyline query* juga dapat dilakukan dengan mengubah *SKYLINE OF* menjadi *nested* SQL *query* seperti yang diilustrasikan pada Gambar 2.3. Namun, komputasi *skyline query* menggunakan *nested loop* (BNL) memiliki kelemahan pada performa, dikarenakan *overhead* yang tinggi akibat kebutuhan evaluasi seluruh kombinasi *tuple* pada *database*.

```
SELECT *
FROM   Hotels h
WHERE  h.city = 'Nassau' AND NOT EXISTS(
       SELECT *
       FROM   Hotels h1
       WHERE  h1.city = 'Nassau' AND
              h1.distance <= h.distance AND
              h1.price <= h.price AND
              (h1.distance < h.distance OR
              h1.price < h.price));
```

Gambar 2.3. Implementasi *Skyline Query* Menggunakan *Nested* SQL *Query* (Borzsony, et al., 2001)

Sebagai contoh implementasi *skyline*, pada Gambar 2.4(a) telah tersedia *dataset* sepuluh rumah (H1 sampai dengan H10) dilengkapi dengan dua atribut, yaitu: harga dan jarak dengan stasiun. Pada umumnya rumah dengan harga yang lebih murah dan lebih dekat dari stasiun merupakan pilihan yang paling banyak diinginkan atau diminati oleh pelanggan. Namun, berdasarkan *dataset* yang tersedia dapat dilihat bahwa dua atribut tersebut nilainya saling berlawanan, dimana semakin dekat jarak rumah dengan stasiun maka harga akan semakin mahal. Maka dari itu, tiap *customer* kemudian akan memerlukan rasio *money-to-value* sesuai dengan preferensi yang diinginkan, dan tiap *customer* tentunya akan memiliki nilai preferensi yang berbeda-beda.

House	price (in thousand €)	Distance (m)
H1	100	1500
H2	1400	500
H3	700	600
H4	1300	1000
H5	900	1300
H6	1600	100
H7	400	300
H8	200	1200
H9	1000	200
H10	500	1400
H11	500	900



(a) (b)

Gambar 2.4. Ilustrasi Implementasi *Skyline Query*. *Dataset* Rumah dengan Atribut Harga dan Jarak dari Stasiun (a), dan Hasil *Skyline Query* (b) (Kalyvas & Tzouramanis, 2018)

Berdasarkan Gambar 2.4(b), H2, H3, H4, H5, H10, dan H11 bukan merupakan anggota *skyline*, karena nilainya tidak lebih baik dari salah satu atau beberapa anggota *skyline*, yaitu: H1, H6, H7, H8, dan H9 yang ditunjukkan pada garis berwarna merah. Sebagai contoh, H2 bukan merupakan anggota *skyline* karena memiliki harga yang lebih mahal daripada H9, dan memiliki jarak ke stasiun lebih jauh daripada H6. Untuk dapat menjadi anggota *skyline*, suatu *data point* setidaknya harus memiliki nilai yang lebih baik pada salah satu dimensi data, atau lebih baik pada seluruh dimensi data. Karena pada contoh kasus ini "lebih baik" didefinisikan sebagai fungsi minimal pada *skyline*, maka *data point* yang memiliki harga lebih rendah dibandingkan dengan *data point* lainnya dan memiliki jarak semakin dekat dengan stasiun merupakan hasil *skyline*.

2.1.2 SaLSa (*Sort and Limit Skyline Algorithm*)

SaLSa merupakan salah satu algoritma dasar komputasi *skyline query* yang diusulkan pada penelitian (Bartolini, et al., 2006). Salah satu kelebihan dari algoritma ini adalah tidak diperlukannya evaluasi keseluruhan *dataset* untuk menentukan hasil *skyline query* dengan membatasi data yang harus di

evaluasi menggunakan nilai *threshold* tertentu. Komputasi SaLSa terdiri dari dua bagian, yaitu: *sorting dataset* dan penentuan *threshold*.

Proses *sorting* diawali dengan melakukan normalisasi data sehingga data akan berada pada *range* [0,1] dengan menggunakan Persamaan (2.1). Data pada dimensi i direpresentasikan sebagai $(p.d_i)$, min_i adalah nilai minimum pada dimensi i, dan max_i adalah nilai maksimum pada dimensi i. Setelah data dinormalisasi, dilakukan proses *sorting* data dengan memperhatikan dua parameter yang berbeda, yaitu: nilai $fmin(p)$ atau nilai minimal dari data p pada seluruh dimensi i dan $sum(p)$ atau jumlah nilai p pada seluruh dimensi i. Dengan terdapatnya dua parameter *sorting*, maka jika ditemukan dua atau lebih nilai $fmin(p)$ yang sama, maka akan diperhatikan parameter kedua yaitu $sum(p)$.

$$f(p.d_i) = \frac{(p.d_i - min_i)}{(max_i - min_i)} \tag{2.1}$$

$$p_{stop} = \max_{i \in d}\{p.d_i\} \tag{2.2}$$

Dataset yang telah di*sorting* kemudian akan dicek satu persatu untuk menemukan hasil *skyline*-nya. Data pertama yang dibaca secara otomatis akan masuk ke *skyline point list* S, kemudian data selanjutnya akan dicek nilainya menggunakan Persamaan (2.3). Jika data p nilainya tidak didominasi oleh seluruh anggota s pada *skyline point list* S, maka data p kemudian akan masuk ke *skyline point list* S. Data p dikatakan tidak didominasi data s jika memenuhi salah satu keadaan pada Persamaan (2.4), yaitu jika data p memiliki nilai kurang dari data s pada salah satu dimensi datanya atau memiliki nilai kurang dari data s pada seluruh dimensi datanya.

Proses evaluasi tidak dilakukan pada keseluruhan *dataset*. *Threshold* (P_{stop}) ditentukan sebagai parameter untuk menghentikan pengecekan data. Setiap data yang masuk sebagai *skyline point list* akan melakukan *update* nilai

P_{stop}, yang nilainya ditentukan menggunakan Persamaan (2.2). Sebelum melakukan cek dominasi data, terlebih dahulu dilakukan pengecekan *threshold*, jikan nilai $fmin(p) \geq P_{stop}$, maka proses evaluasi data dihentikan. Algoritma komputasi SaLSa *skyline* dapat dilihat pada Gambar 2.5.

$$p \in S, \text{ if } \nexists\, s \in S \text{ such that } s \prec p \tag{2.3}$$

$$s \prec p, \text{if (a) } s_i > p_i, \forall i \in [1, \dots d] \text{ and (b) } s_i > p_i, \exists i \in [1, \dots d] \tag{2.4}$$

Algorithm 1 SaLSa

1: $\mathcal{S} \leftarrow \emptyset, \mathcal{U} \leftarrow r$, $stop \leftarrow false$, $p_{stop} \leftarrow$ undefined
2: sort $\mathcal{U}$ according to $\mathcal{F}$
3: **while not** $stop \wedge \mathcal{U} \neq \emptyset$ **do**
4: $\quad p \leftarrow$ get next point from $\mathcal{U}$, $\mathcal{U} \leftarrow \mathcal{U} \setminus \{p\}$
5: $\quad$ **if** $\mathcal{S} \not\succ p$ **then** $\mathcal{S} \leftarrow \mathcal{S} \cup \{p\}$, update p_{stop}
6: $\quad$ **if** $p_{stop} \succ \mathcal{U}$ **then** $stop \leftarrow true$
7: return $\mathcal{S}$

Gambar 2.5. Algoritma SaLSa *Skyline*
(Bartolini, et al., 2006)

Tabel 2.1 merupakan ilustrasi komputasi *skyline* menggunakan algoritma SaLSa. Berdasarkan hasil *sorting*, diperoleh urutan pertama dari *dataset* adalah H1. Karena H1 berada pada urutan pertama dan belum terdapat *skyline point* sebelumnya, maka H1 merupakan anggota pertama *skyline*. Dengan masuknya H1 sebagai anggota *skyline*, nilai P_{stop} akan di*update* menjadi 1, yang merupakan nilai data tertinggi pada seluruh dimensi data H1.

Kemudian, H6 akan dibandingkan dengan H1. Karena H6 tidak terdominasi oleh H1 (karena jarak H6 lebih baik daripada H1), H6 akan masuk menjadi anggota *skyline* dan P_{stop} akan tetap bernilai 1 karena nilai data tertinggi pada keseluruhan dimensi data H6 adalah 1. Data selanjutnya pada

dataset adalah H8 yang juga tidak terdominasi oleh *data point* lain anggota *skyline*, sehingga H8 juga akan masuk menjadi anggota *skyline* dan menyebabkan nilai P_{stop} berubah menjadi 0,786. H9 dan H7 juga merupakan *data point* yang tidak terdominasi, dan merupakan anggota *skyline*. Dengan masuknya H7 sebagai anggota *skyline*, nilai akhir P_{stop} adalah 0,200.

Saat H11 masuk sebagai *data point* yang akan dievaluasi selanjutnya, teridentifikasi bahwa nilai $fmin$-nya melebihi nilai P_{stop}, sehingga komputasi *skyline* dihentikan dan menghasilkan H1, H6, H8, H9 dan H7 sebagai anggota *skyline*. Berdasarkan ilustrasi berikut dapat dibuktikan pula bahwa penggunaan algoritma SaLSa untuk komputasi *skyline* tidak memerlukan proses evaluasi terhadap keseluruhan data yang ada pada *dataset*.

Tabel 2.1. Ilustrasi Penentuan *Skyline* Menggunakan Algoritma SaLSa pada *Dataset* Rumah (Kalyvas & Tzouramanis, 2018)

House	Price	Distance	$f_{min}(h)$	Sum(h)	p_i	P_{stop}
H1	0	1	0	1	1	1
H6	1	0	0	1	1	1
H8	0,067	0,786	0,067	0,853	0,786	0,786
H9	0,600	0,071	0,071	0,671	0,600	0,600
H7	0,200	0,143	0,143	0,343	0,200	0,200
H11	0,267	0,571	0,267	0,838	-	Stop! $F_{min}(H_{11}) \geq P_{stop}$
H10	0,267	0,929	0,267	1,196	-	
H2	0,867	0,286	0,286	1,153	-	
H3	0,400	0,357	0,357	0,757	-	
H5	0,530	0,857	0,533	1,387	-	
H4	0,800	0,643	0,643	1,443	-	

2.1.3 *Dynamic Skyline Query* (DSL)

Salah satu variasi dari *skyline query* pada penelitian (Papadias, et al., 2003) adalah *dynamic skyline query*. DSL akan membentuk *data space* baru sesuai dengan nilai *query* yang akan dievaluasi, dengan menghitung *dynamic coordinate* dari setiap data *original* pada *original data space* menggunakan Persamaan (2.5). Hasil DSL dapat ditentukan dengan melakukan pengecekan *dynamic dominance* terhadap setiap data pada *data space*, dimana p dikatakan mendominasi r *with respect to* (w.r.t) q apabila salah satu keadaan pada Persamaan (2.6) terpenuhi.

$$p_i' = |p_i - q_i| + q_i \tag{2.5}$$

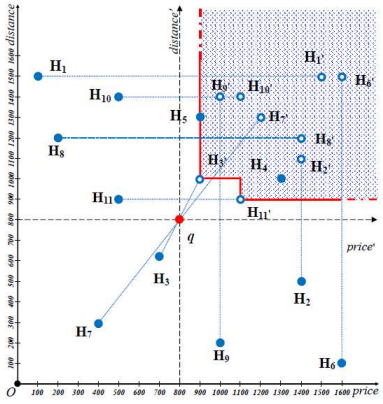
$p \prec^q r$, if

(a) $\exists j \in [1, d]$ such that $\left|q_{(d_i)} - p_{(d_i)}\right| < \left|q_{(d_i)} - r_{(d_i)}\right|$ or

(b) $\forall j \in [1, d]$,$\left|q_{(d_i)} - p_{(d_i)}\right| < \left|q_{(d_i)} - r_{(d_i)}\right|$ (2.6)

DSL *query result* akan berisi *data point* yang nilainya memenuhi Persamaan (2.7). Ilustrasi perbedaan antara *skyline* dan *dynamic skyline* dapat dilihat pada Gambar 2.6. Tiap *data point* akan berada pada posisi baru sesuai dengan nilai *query* q yang dievaluasi. Pada komputasi *skyline* yang digunakan sebagai evaluasi *customer-product relationship*, DSL biasanya digunakan pada perspektif *customer*, sehingga nilai q akan mewakili nilai preferensi dari tiap *customer*. Hasil komputasi *skyline* terhadap tiap *customer* yang berbeda tentunya akan berbeda pula, hal ini dikarenakan tiap *customer* tentunya akan memiliki nilai preferensi yang berbeda satu sama lain.

$$DSL_{(q)}(Ds) = \left\{p \in Ds \mid \nexists r \in Ds : r \prec_{Ds}^{q} p\right\} \tag{2.7}$$



Gambar 2.6. Ilustrasi *Dynamic Skyline* pada *Dataset* Rumah (Kalyvas & Tzouramanis, 2018)

2.1.4 *Reverse Skyline Query* (RSL)

Reverse skyline (Dellis & Seeger, 2007) merupakan kebalikan dari DSL. *Query* RSL ($RSL_{(q)}$) akan menampilkan *data point* pada *dataset* yang memunculkan *query* q sebagai hasil DSL-nya, dimana RSL *query result* akan

berisi *data point* yang nilainya memenuhi Persamaan (2.8). Pada evaluasi *customer-product relationship*, RSL digunakan untuk mengetahui *customer* mana saja yang tertarik terhadap produk p seperti pada penelitian (Wu, et al., 2009) dan (Islam & Liu, 2016).

$$RSL_{(q)}(Ds) = \{p \in Ds \mid \nexists r \in Ds : r \prec_{Ds}^{q} p\} \tag{2.8}$$

2.2. Studi Literatur

2.2.1 *K-Most Promising Product* ($K - MPP$)

Mengetahui *most promising product* (produk potensial yang akan dipilih oleh *customer* daripada produk lain) di dunia perdagangan merupakan hal yang sangat penting. Dengan tren penjualan secara *online* seperti saat ini, mengetahui *most promising product* untuk evaluasi produsen bukanlah hal yang susah, karena produsen dapat dengan mudah mengetahui *preference* produk yang diinginkan oleh *customer* dengan menggunakan *search queries* dari *online user*. Untuk menentukan *most promising product*, akan dilakukan evaluasi *product adoption* pada *dataset preference customer* dan *product selection* pada *dataset* produk. Dengan menggabungkan kedua model evaluasi tersebut, dapat diketahui *most promising product* dengan menggunakan *framework* yang diberi nama $K - MPP$ oleh penulis.

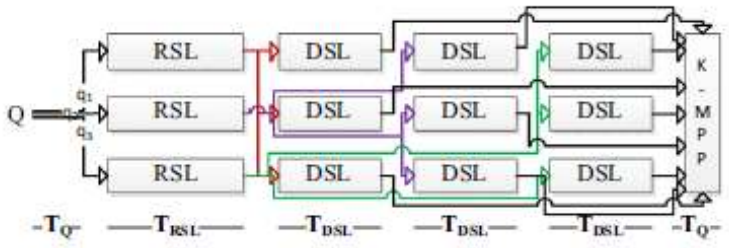
Metode komputasi yang digunakan untuk menentukan *product adoption* adalah *skyline operator* (Borzsony, et al., 2001). Pada *product adoption* dilakukan komputasi *dynamic skyline* (Papadias, et al., 2003) untuk menentukan produk yang nilainya tidak didominasi oleh produk lain menggunakan *dataset customer preference*, dan komputasi *reverse skyline* (Dellis & Seeger, 2007) untuk menentukakn *customer* yang *preference* nya sesuai dengan *query* produk dan nilainya tidak didominasi oleh *customer* lain. Langkah selanjutnya adalah menentukan nilai *probability* dari tiap *customer* menggunakan hasil komputasi DSL pada *product adoption* (probabilitas dipilihnya suatu produk oleh *customer*) yang dinotasikan sebagai $Pr(c, p|P)$

menggunakan Persamaan (2.9). Semakin banyak produk yang disukai oleh seorang *customer*, maka nilai *probability* dari *customer* tersebut akan semakin kecil. Kemudian, dapat dihitung nilai *market contribution* suatu produk menggunakan Persamaan (2.10). Nilai *market contribution* dianggap sebagai nilai yang mewakili jumlah skor suatu produk yang nantinya akan dibandingkan dengan produk lain untuk mengetahui *most promising product*. Skor tiap produk akan diperoleh dengan menjumlahkan nilai *probability* dari tiap *customer* yang muncul pada hasil RSL tiap produk.

$$Pr(c,\ p|P) = \begin{cases} \dfrac{1}{|DSL_{(c)}|} & \text{jika } p \\ 0 & \end{cases} \in\ DSL_{(c)} \tag{2.9}$$

$$E(C,p|P) = \sum_{\forall_c \in C} Pr(c,\ p|P) \tag{2.10}$$

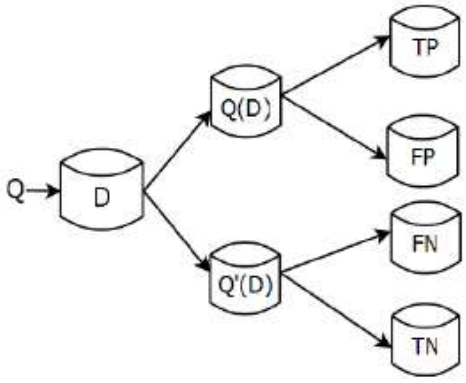
Berdasarkan model *product adoption* diatas, digunakan model *product selection* untuk mencari K-produk yang memiliki nilai *market contribution* terbesar yang diberi nama *K-most promising product* ($K - MPP$). Untuk efisiensi pencarian $K - MPP$, diusulkan juga komputasi *skyline query* secara pararel yang dilustrasikan pada Gambar 2.7.



Gambar 2.7. Ilustrasi Proses Penentuan $K - MPP$ dengan Komputasi Pararel (Islam & Liu, 2016)

2.2.2 *Why* dan *Why-not Questions* pada *Database*

Penelitian (Islam, 2013) mengusulkan *framework* FlexIQ yang menggunakan *user feedback* untuk menyelesaikan permasalahan *why* dan *why-not question* pada *Select- Project-Join* (SPJ) *query result* dengan melakukan modifikasi *query point*. *Feedback why question* akan berisi *unexpected data* yang muncul pada *query result*, sedangkan *why-not question* berisi *expected data* yang tidak muncul pada *query result*. Setelah mendapatkan *feedback*, dilakukan pengelompokann *tuple database* menjadi empat kaegori, yaitu: (i) *truly positive tuples* $(TP) \subseteq Q(D)$, (ii) *false positive tuples* $(FP) \subseteq Q(D)$, (iii) *truly negative tuples* $(TN) \subseteq Q'(D)$, dan (iv) *false negative tuples* $(FN) \subseteq Q'(D)$ seperti yang diilustrasikan pada Gambar 2.8.

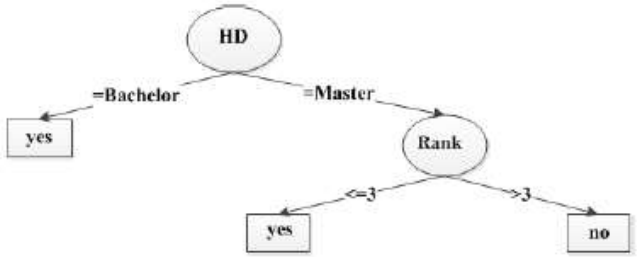


Gambar 2.8. Taksonomi *Tuple Database* w.r.t *Query q*
(Islam, 2013)

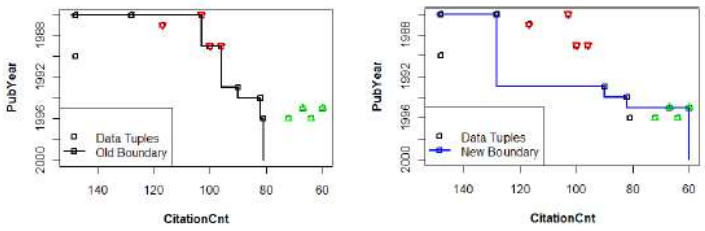
Setelah proses pengelompokan, kemudian akan dibuat *feedback table* yang berisi data dari *tuple* yang berada pada kelompok TP, FP, dan FN dan ditambahkan juga kolom *feedback* yang berisi keterangan '*yes'* atau '*no*' berdasarkan *user feedback*. Selanjutnya, berdasarkan *feedback table* yang telah dibuat, digunakan *decision-tree* (DT) *classifier* untuk mengetahui kriteria *data refinement* yang diinginkan oleh *user* (Gambar 2.9) . Langkah selanjutnya melibatkan *skyline operator* untuk menentukan *data refinement*, yaitu dengan mengubah *original query result boundary* menjadi *query result boundary* baru sesuai dengan *user feedback,* seperti yang diilustrasikan pada

Gambar 2.10, dimana *unexpected data* direpresentasikan sebagai segitiga berwarna merah dan *expected data* segitiga berwarna hijau.

Sname	HD	Npub	Rank	Feedback
John	Bachelor	1	1	yes
Peter	Master	2	4	no
Noah	Master	2	3	yes



Gambar 2.9. *Feedback Table* dan *Decision Tree* (DT) *Classifier*
(Islam, 2013)



Gambar 2.10. *Original Query Result Boundary* (a) dan *Query Result Boundary* Baru dan *Data Point* (b)
(Islam, 2013)

Untuk menentukan *query result boundary* yang baru, langkah pertama yang dilakukan adalah mengubah *boundary* dengan mempertimbangkan *unexpected data*. Jika terdapat *unexpected data* yang nilainya mendominasi *original query result boundary*, maka perlu dilakukan perubahan *boundary* hingga tidak ada lagi *unexpected data* yang mendominasi *original query result boundary*. Selanjutnya, dilakukan perubahan *boundary* dengan mempertimbangkan *expected data*. Jika terdapat *expected data* yang nilainya terdominasi oleh *original query result boundary*, maka perlu dilakukan

perubahan *boundary* hingga tidak ada lagi *expected data* yang didominasi oleh *original query result boundary*.

New query result boundary yang telah ditentukan kemudian dijadikan dasar penentuan *data refinement*. Terdapat dua metode yang diusulkan untuk menentukan *data refinement*. Metode pertama diberi nama *baseline algorithm* (TBA), dimana *data refinement* akan dibentuk menggunakan fungsi konjungsi data pada *database*. Contoh dari penggunaan TBA pada *new query result boundary* pada Gambar 2.10(b) adalah sebagai berikut: SELECT pubidFROM publication WHERE ((citationcnt ≥ 128 AND pubyear ≥ 1986)OR (citationcnt ≥ 90 AND pubyear ≥ 1993) OR (citationcnt ≥ 82 AND pubyear ≥ 1994) OR (citationcnt ≥ 60 AND pubyear ≥ 1995)).

Metode usulan kedua diberi nama *trade-off algorithm* (TOA), dimana akan dibuat beberapa *subquery* dengan menggabungkan beberapa *query* untuk efisiensi. Dengan menggunakan TOA, maka contoh *data refinement* yang dihasilkan adalah: SELECT pubid FROM publication WHERE ((citationcnt ≥ 128 ANDpubyear ≥ 1986) OR (citationcnt ≥ 60 AND pubyear ≥ 1993)).

2.2.3 *Why-not Questions* pada *Reverse Skyline Query*

Untuk menjawab *why-not question pada reverse skyline query*, (Islam, et al., 2013) mengusulkan tiga solusi yang berbeda, yaitu: modifikasi *data point*, modifikasi *query*, serta modifikasi *data point* & *query*. Untuk solusi modifikasi *query* diusulkan dua metode, yaitu: tanpa memperhatikan *safe region* dan dengan memperhatikan *safe region*. Dengan menggunakan metode kedua, yaitu melakukan perubahan *query* pada *safe region*, maka $c_l \epsilon RSL_{(q)}$ juga akan menjadi anggota $RSL_{(q')}$. Tujuan dilakukan penentuan *safe region* adalah untuk meminimalisir perubahan anggota RSL saat dilakukan perubahan nilai *query*. Modifikasi *query* akan dilakukan dengan mengubah nilai q menjadi q' sehingga *why-not point* c_t muncul sebagai anggota $RSL_{(q')}$.

Modifikasi *Query* tanpa Mempertimbangkan *Safe Region*

Identifikasi *data point* yang mendominasi q sehingga c_t tidak muncul sebagai $RSL_{(q)}$ dapat dilakukan dengan membuat *window query*, sebagai contoh: $\Lambda \longleftarrow window\_query(c_t, q)$. *Window query* dapat direpresentasikan sebagai persegi atau persegi panjang yang diagonalnya menghubungkan c_t dan q. Kemudian, dilakukan komputasi F, yang merupakan kumpulan *data point* yang muncul pada Λ dan $DSL_{(c_t)}$. Langkah pertama yang akan dilakukan untuk menentukan F adalah inisialisasi, $F \longleftarrow \Lambda$, kemudian dilakukan cek *pairwise dominance* terhadap seluruh $e_i \in F$ menggunakan Persamaan (2.11).

$$\text{if for each } e_1 \in F, \exists e_2 \in F \text{ such that } e_1 \succ_{c_t} e_2 \text{ then remove } e_2 \text{ from } F \tag{2.11}$$

Untuk menentukan nilai q', akan dicari nilai maksimal data F di tiap dimensinya. Kemungkinan perubahan nilai q' dengan *cost* perubahan yang minimal kemudian diperoleh dengan mengubah data q pada tiap dimensi $[i,..d]$ dengan nilai maksimal dari tiap dimensi $[i,..d]$ pada data F, misal: $[q^1, F^1_{max}]$ dan $[F^2_{max}, q^2]$.

Modifikasi *Query* dengan Mempertimbangkan *Safe Region*

Pada metode penentuan perubahan q' yang mempertimbangkan *safe region*, akan terlebih dahulu ditentukan *dynamic anti dominance area* ($\overline{DDR}$) dari setiap $c_l \epsilon RSL_{(q)}$ dengan melakukan *pairwise check*. *Dynamic anti dominance area* adalah area yang tidak didominasi oleh DSL *query result* yang merupakan area dibawah dan disamping garis *skyline result*. Langkah penentuan *safe region* dapat dilihat pada Gambar 2.11. Dengan ditentukannya *safe region* yang merupakan perpotongan atau irisan dari $\overline{DDR}$ anggota $RSL_{(q)}$, selama q' berada pada area *safe region*, maka anggota $RSL_{(q)}$ juga akan menjadi anggota $RSL_{(q')}$ bersama dengan *data point* baru lainnya.

Algorithm 3 Exact Safe Region ($RSL(q)$)

1: $SR(q) \leftarrow null$
2: **for** each $c_l \in RSL(q)$ **do**
3: Compute $\overline{DDR}(c_l)$;
4: **if** $SR(q) = null$ **then**
5: $SR(q) \leftarrow \overline{DDR}(c_l)$;
6: **else**
7: $SR(q) \leftarrow SR(q) \cap \overline{DDR}(c_l)$;

Gambar 2.11. Algoritma untuk Penentuan *Safe Region*
(Islam, et al., 2013)

2.2.4 *Why-not Questions* pada *Reverse Top-k Query*

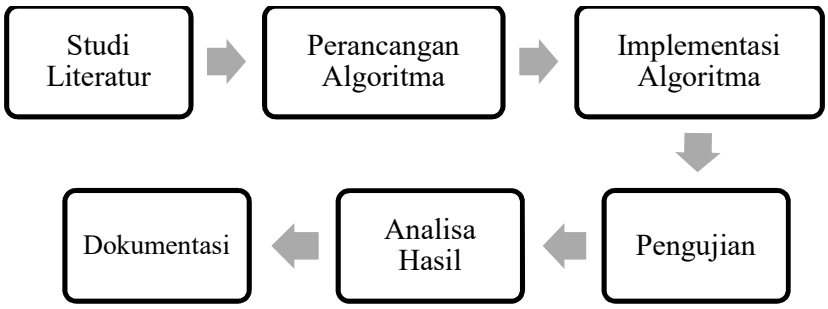
Penelitian (Liu, et al., 2016) membahas mengenai langkah penyelesaian *why-not* dan *why question* pada *reverse top-k query*. Metode yang digunakan untuk menjawab *why-not question* yang digunakan pada penelitian ini hampir sama dengan penelitian (Islam, et al., 2013), dimana dilakukan pula modifikasi *data point* (dalam *top-k query* disebut *weight*), modifikasi *query*, serta modifikasi *weight*, *query* dan k. Proses modifikasi *query* dilakukan dengan dua metode yang berbeda, yaitu dengan mempertimbangkan *safe region* dan dengan menggunakan fungsi *quadratic programming* pada Persamaan (2.12). Liu et al (2016) pada penelitiannya mengatakan bahwa penentuan modifikasi *query* dengan mempertimbangkan *safe region* memiliki kelemahan pada data dengan dimensionalitas tinggi, maka dari itu diusulkan penggunaan *quadratic programming* untuk efisiensi.

$$\min f(x) = \frac{1}{2}(q')^T H q' + (q')^T c$$
$$s.t. \begin{cases} Aq' \leq b \\ 0 \leq q' \leq q \end{cases} \tag{2.12}$$

BAB 3
METODOLOGI PENELITIAN

Bab ini menjelaskan mengenai tahapan penelitian yang dibutuhkan agar tujuan penelitian tercapai. Tahap pertama yang dilakukan dalam penelitian ini adalah studi literatur. Pada penelitian ini, studi literatur dilakukan untuk mengumpulkan informasi mengenai *skyline query* dan identifikasi permasalahan pada penelitian-penelitian yang pernah dilakukan sebelumnya. Setelah dilakukan tahap studi literatur, diperoleh penelitian atau kontribusi yang akan dikembangkan selanjutnya pada penelitian ini, yaitu perumusan algoritma yang dapat memberikan solusi informatif apabila muncul *why-not question* pada hasil *query* $K - MPP$.

Tahap selanjutnya adalah perancangan dan implementasi algoritma. Setelah menemukan topik penelitian yang akan dikembangkan pada tahap pertama, ditentukan ide yang akan diusulkan dan solusi yang ditawarkan untuk menyelesaikan permasalahan tersebut. Pada penelitian ini, algoritma yang dirancang menggunakan pendekatan *data refinement* untuk menjawab *why-not question* pada $K - MPP$. Ide dan solusi yang diusulkan kemudian akan diimplementasikan dan dianalisis pada beberapa skenario uji yang berbeda dan hasil yang diperoleh akan dievaluasi untuk mendapatkan kesimpulan mengenai penerapan algoritma untuk menyelesaikan permasalahan *why-not* pada $K - MPP$. Dokumentasi kemudian akan dituliskan dalam bentuk buku Tesis. Ilustrasi alur metodologi penelitian dapat dilihat pada Gambar 3.1.



Gambar 3.1. Alur Metodologi Penelitian

3.1 Studi Literatur

Penelitian diawali dengan melakukan kajian yang berkaitan dengan topik penelitian yang diajukan. Referensi yang digunakan dalam penelitian ini berasal dari jurnal, konferensi, dan buku yang berkaitan dengan *skyline query* dan implementasinya dalam proses penentuan *query result* pada sistem *database* serta penyelesaian permasalahan *why-not question* pada *query result*. Berdasarkan studi literatur yang telah dilakukan, diperoleh informasi sebagai berikut.

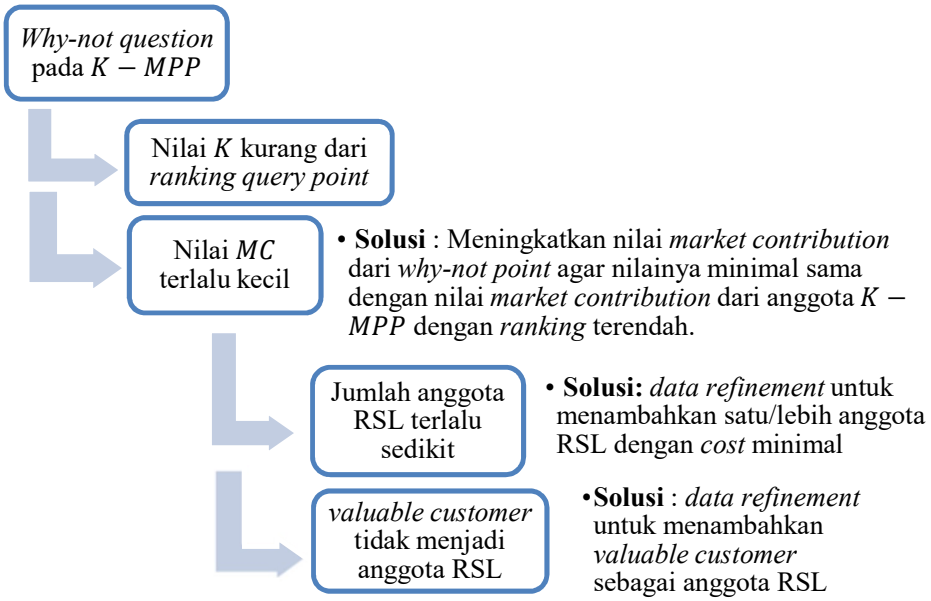
1. Terdapat beberapa jenis algoritma *skyline* dasar, diantaranya: D&C, Bitmap, Index, NN, BBS, BNL, SFS, LESS, dan SaLSa
2. Komputasi *skyline* dapat dikelompokkan menjadi dua kategori berdasarkan dibutuhkan atau tidak *pre-processing* untuk *data indexing*. Kategori yang pertama adalah *Index-based skyline*, yang memiliki performa lebih baik dibandingkan dengan kategori kedua: *non idex-based skyline*, karena dengan melakukan *indexing* maka tidak diperlukan pengecekan data satu persatu. Salah satu permasalahan pada *index-based skyline* adalah *curse of dimensionality* yaitu timbulnya kemungkinan penurunan performa seiring dengan jumlah dimensi data. *Non index-based skyline* tidak membutuhkan struktur khusus untuk komputasi *skyline* dan lebih umum digunakan.
3. *K-Most Promising Product* ($K - MPP$) dapat digunakan untuk menentukan strategi marketing (menentukan *K*-produk yang paling diminati oleh *customer*) dengan melakukan evaluasi terhadap *dataset customer* dan produk menggunakan *skyline query*.
4. Jika *query result* pada $K - MPP$ kurang sesuai dengan keinginan *user*, belum disediakan solusi informatif dan analisis hasil untuk memudahkan *user* yang kurang memahami permasalahan pencarian *query* dan sistem *database*.
5. Untuk menjawab *why-not question* atau *why-not query result* pada pencarian sistem *database*, terdapat tiga klasifikasi metode penelitian yang umumnya dilakukan, yaitu: mengidentifikasi penyebab suatu *data point* tidak muncul sebagai *query result*, modifikasi *data point* pada sistem *database* sehingga *why-not point* dapat muncul sebagai *query result*, dan modifikasi *query* sehingga *why-not point* dapat muncul sebagai *query result*.

3.2 Perancangan Algoritma

Alur perancangan algoritma dibagi menjadi empat tahap, yaitu: persiapan, pendefinisian atribut dan pendefinisian metode yang diusulkan.

3.2.1 Persiapan

Pada tahap ini dilakukan identifikasi permasalahan yang menyebabkan munculnya *why-not question* pada $K - MPP$. Daftar identifikasi permasalahan dapat dilihat pada Gambar 3.2. Data identifikasi permasalahan dapat digunakan pada langkah penelitian selanjutnya, yaitu pendefinisian metode yang diusulkan. Identifikasi masalah dilakukan dengan menganalisis proses dan hasil pencarian $K - MPP$ pada penelitian sebelumnya.



Gambar 3.2. *Breakdown* Identifikasi Permasalahan dan Solusi yang Mungkin Dilakukan terhadap Munculnya *Why-not Question* pada $K - MPP$ *query result*

3.2.2 Pendefinisian Atribut

Bagian ini mendefinisikan atribut-atribut yang terlibat dalam penelitian yang akan dilakukan. Daftar atribut dapat dilihat pada Tabel 3.1. Pendefinisian

atribut hanya sebatas identifikasi nama atau simbol, dan definisi dari atribut tersebut.

Tabel 3.1. Daftar Atribut

Simbol	Definisi
C	Himpunan (set) *customer*
P	Himpunan (set) produk
PP	Himpunan (set) produk anggota $K-MPP$
c	*Customer*
p	Produk
pp	Produk yang merupakan anggota $K-MPP$
$DSL(c)$	*Dynamic Skyline customer*
$RSL(p)$	*Reverse Skyline* produk
$NRSL_{(p)}$	Jumlah angota $RSL(p)$
$Pr(c,p\|P)$	Probabilitas *customer* (c) memilih produk (p)
$MC(C,p\|P)$	*Market contribution* produk
$K-MPP$	*K-most promising product*
$nK-MPP$	Non K-*most promising product*
K'	*K refinement*
MVC	*Most Valuable Customer*
q	*Query point* / *why-not point*
q'	*Data refinement*
$list\_new\_data$	Himpunan kemungkinan q'
ΔK	Selisih nilai K pada $K-MPP$ dan K'

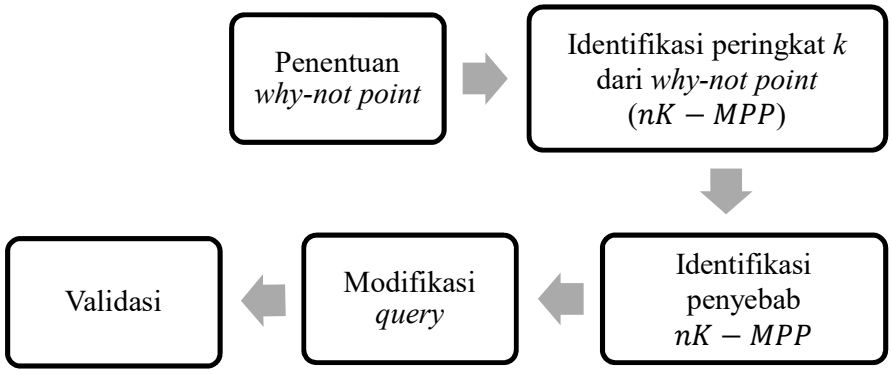
3.2.3 Pendefinisian Metode yang Diusulkan

Berdasarkan hasil *breakdown* identifikasi permasalahan yang telah dilakukan pada tahap sebelumnya, dapat disimpulkan bahwa langkah untuk menjawab *why-not* $K-MPP$ terdiri dari beberapa tahapan yang secara keseluruhan dapat dilihat pada Gambar 3.3. Tahap utama dalam proses menjawab *why-not question* pada $K-MPP$ diantaranya terdiri dari:

identifikasi peringkat K dari produk $nK - MPP$, meningkatkan nilai *market contribution* dengan melakukan modifikasi nilai *query* yang merupakan *why-not point* (*data refinement*), dan validasi.

Sebelum melakukan modifikasi *query*, terlebih dahulu perlu dilakukan identifikasi penyebab *why-not point* tidak muncul sebagai anggota $K - MPP$. Setelah penyebab tidak munculnya *why-not point* pada $K - MPP$ ditemukan, kemudian proses modifikasi *query* atau *data refinement* dapat dilakukan dengan melakukan evaluasi pada list *data point* yang muncul sebagai anggota $K - MPP$. Proses modifikasi *query* kemudian akan menghasilkan beberapa kombinasi kemungkinan *data refinement* pada salah satu dimensi data *why-not point*.

Setelah diperoleh daftar kombinasi kemungkinan *data refinement* kemudian dilakukan proses validasi untuk menjamin kebenaran solusi *data refinement* yang diberikan. Proses validasi dilakukan dengan melakukan pengecekan apakah solusi *data refinement* yang diberikan dapat menjadikan *why-not point* menjadi salah satu anggota $K - MPP$.



Gambar 3.3. Alur Perancangan dan Penerapan Algoritma untuk Menjawab *Why-Not Question* pada $K - MPP$

Penentuan *why-not-point*

Pada penelitian ini, *why-not question* diilustrasikan sebagai keadaan dimana *user* (produsen) kurang puas dengan hasil *query* $K - MPP$ karena produknya bukan merupakan *top K-promising product*. Produk yang ingin

dievaluasi karena tidak muncul sebagai hasil $K - MPP$ tersebut kemudian disebut sebagai *why-not point*. Maka dari itu, *why-not point* merupakan *user feedback* yang nantinya akan dievaluasi pada tahapan berikutnya.

Identifikasi Peringkat *Why-Not Point* ($nK - MPP$)

Identifikasi peringkat dari *why-not point* merupakan langkah awal untuk mengetahui peringkat suatu produk $nK - MPP$ terhadap keseluruhan produk yang terdapat pada *dataset*. Karena $K - MPP$ hanya menampilkan K- produk dengan nilai *market contribution* terbaik, tahapan ini dilakukan untuk evaluasi nilai *market contribution* dari *why-not point* terhadap keseluruhan produk. Dengan begitu, penyebab suatu produk merupakan anggota *why-not* $K - MPP$ dapat terjawab dengan memberikan solusi informatif pertama berupa informasi *ranking* produk *why-not point* tersebut.

Identifikasi peringkat nilai *market contribution* dari *why-not point* terhadap keseluruhan produk $MC(C, q|P)$ yang selanjutnya dinotasikan sebagai K', dapat ditentukan dengan mengubah nilai K sehingga nilainya sama dengan *ranking* nilai *market contribution* dari *query point* terhadap keseluruhan produk $K' = rank(MC(C, q|P))$. Peringkat K' dan hasil *query* $K' - MPP$ kemudian akan digunakan pada tahapan berikutnya.

Tabel 3.2. *Dataset* Produk P (a) dan Preferensi *Customer* C (b)

(Islam & Liu, 2016)

ID	Dim1	Dim2
p_1	6	6
p_2	4	18
p_3	6	20
p_4	9	15
p_5	12	18
p_6	16	14
p_7	12	6
p_8	16	6
p_9	20	8
p_{10}	20	20

(a)

ID	Dim1	Dim2
c_1	2	8
c_2	4	10
c_3	6	16
c_4	8	18
c_5	10	10
c_6	16	14
c_7	12	2
c_8	18	6
c_9	18	18
c_{10}	20	13

(b)

Tabel 3.3. Nilai *Market Contribution* dari Keseluruhan *Dataset* Produk

Ranking	**Produk**	$\boldsymbol{RSL_{(p)}}$	**MC**
1	p_{11}	c_{10}, c_7, c_5	1,666
2	p_1	c_3, c_8, c_2, c_1	1,583
3	p_5	c_4, c_7, c_5	1,533
4	p_7	c_7, c_8	1,5
	p_8	c_8, c_6	1,5
5	p_{13}	$c_{5,}c_6$	1,333
6	p_9	c_{10}, c_8, c_1	1,166
7	p_6	c_6	1
8	p_2	c_4, c_2, c_3	0,95
9	p_3	c_3, c_4	0,7
	p_4	c_4, c_3	0,7
10	p_{10}	c_9, c_{10}	0,666
	p_4	c_4, c_7, c_5	0,666
11	p_{12}	c_3	0,5

Contoh 1. Berdasarkan *Dataset* pada Tabel 3.2, data nilai *market contribution* pada Tabel 3.3 diperoleh setelah melakukan proses komputasi DSL, RSL dan *probability*. Tiga produk yang memiliki nilai MC terbaik, sekaligus merupakan $3-MPP$ adalah produk $p_{11}, p_1,$ dan p_5. Hanya ketiga produk tersebut yang nantinya akan ditampilkan pada *query result* $K-MPP$. Jika produsen produk p_8 melihat hasil tersebut, maka akan timbul pertanyaan mengapa produknya tidak muncul sebagai hasil $3-MPP$. Oleh karena itu, sebagai solusi informatif pertama akan dicek *ranking* dari produk p_8 dan dijadikan sebagai nilai *K'*, sehingga *K'* = 4. Setelah didapatkan nilai *K'* maka akan ditampilkan *query result* dari $4-MPP$.

Identifikasi penyebab *why-not question*

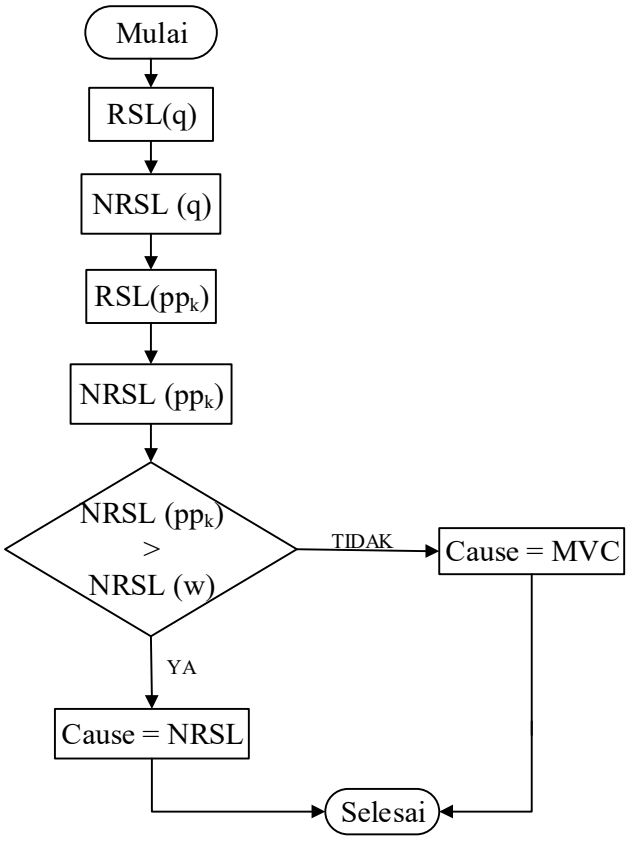
Semakin tinggi nilai *market contribution*, maka semakin besar peluang suatu produk untuk muncul sebagai hasil $K - MPP$. Karena nilai *market contribution* didapatkan dari penjumlahan nilai *probability* produk dari tiap anggota RSL, identifikasi penyebab *why-not question* dapat dilakukan dengan mengevaluasi anggota RSL dari produk $K - MPP$ dan $nK - MPP$ seperti pada diagram alur yang ditunjukkan pada Gambar 3.4

Definisi 1. (Penyebab *why-not question*) Untuk set produk P , *reverse skyline* produk $RSL_{(p_i)}$, jumlah *reverse skyline* produk $NRSL_{(p_i)}$, set *K-most promising product* $K - MPP$, dan set *non K-most promising product* $nK - MPP$.
Penyebab $p_i \in nK - MPP$ dapat diidentifikasi menggunakan $RSL_{(p_i)}$.
(i)$NRSL_{(p_i)} < \min NRSL_{(pp_j)}$, dimana $pp_j \in K - MPP$, maka penyebabnya adalah kurangnya $NRSL_{(p_i)}$.
(ii) $NRSL_{(p_i)} \geq \min NRSL_{(pp_j)}$, maka penyebabnya adalah $MVC \not\ni RSL_{(p_i)}$ atau *ranking* MVC anggota RSL tidak lebih baik daripada *ranking* MVC anggota $K - MPP$.

Berdasarkan proses identifikasi penyebab *why-not question* diatas, untuk meningkatkan nilai *market contribution*, terdapat dua solusi yang mungkin dilakukan, yaitu: (a) menambah jumlah anggota RSL, dan (b) menambah anggota RSL dengan c yang memiliki nilai *probability* K-terbesar, yang selanjutnya disebut sebagai *most valuable customer*.

Definisi 2. (*Most Valuable Customer*) Untuk set *customer* C, *dynamic skyline customer* $DSL_{(c)}$, dan *probability customer* c memilih produk p $\Pr(c, p|P)$. *Most valuable customer* terdiri dari *customer* c yang nilai *probability*-nya K-terbesar $MVC = K - \Pr(c, p|P)$, dimana nilai K untuk proses identifikasi MVC sama dengan nilai K yang didefinisikan pada $K - MPP$.

Contoh 2. Berdasarkan Tabel 3.4, dapat dilihat bahwa p_7 dan p_8 tidak muncul sebagai $3-MPP$ karena $NRSL_{(p_7)}$ dan $NRSL_{(p_8)}$ kurang dari $NRSL_{(p_{11})}$ dan $NRSL_{(p_5)}$ yang memiliki jumlah RSL minimal diantara seluruh anggota $3-MPP$.



Gambar 3.4. Diagram Alur Proses Identifikasi Penyebab *Why-Not Question* pada $K-MPP$

Contoh 3. p_9 memiliki jumlah anggota RSL sama dengan p_{11} dan p_5, namun tidak muncul sebagai $3-MPP$ karena c_7 yang memiliki nilai *probability* terbaik pada $3-MVP$ bukan merupakan anggota $RSL_{(p_9)}$.

Tabel 3.4. Nilai *Probability* Produk dari Keseluruhan *Dataset Customer*

Ranking	***Customer***	$\boldsymbol{DSL_{(c)}}$	***Probability***
1	c_6	p_6	1
	c_7	p_7	1
2	c_3	p_{12}, p_3	0,5
	c_8	p_{14}, p_8	0,5
3	c_1	p_2, p_9, p_1	0,333
	c_5	p_{13}, p_4, p_{11}	0,333
	c_9	p_5, p_{10}, p_{14}	0,333
	c_{10}	p_6, p_9, p_{14}	0,333
4	c_2	$p_{13}, p_2, p_{11,} p_1$	0,25
5	c_4	$p_4, p_{12}, p_2, p_3, p_5$	0,2

Modifikasi *query point*

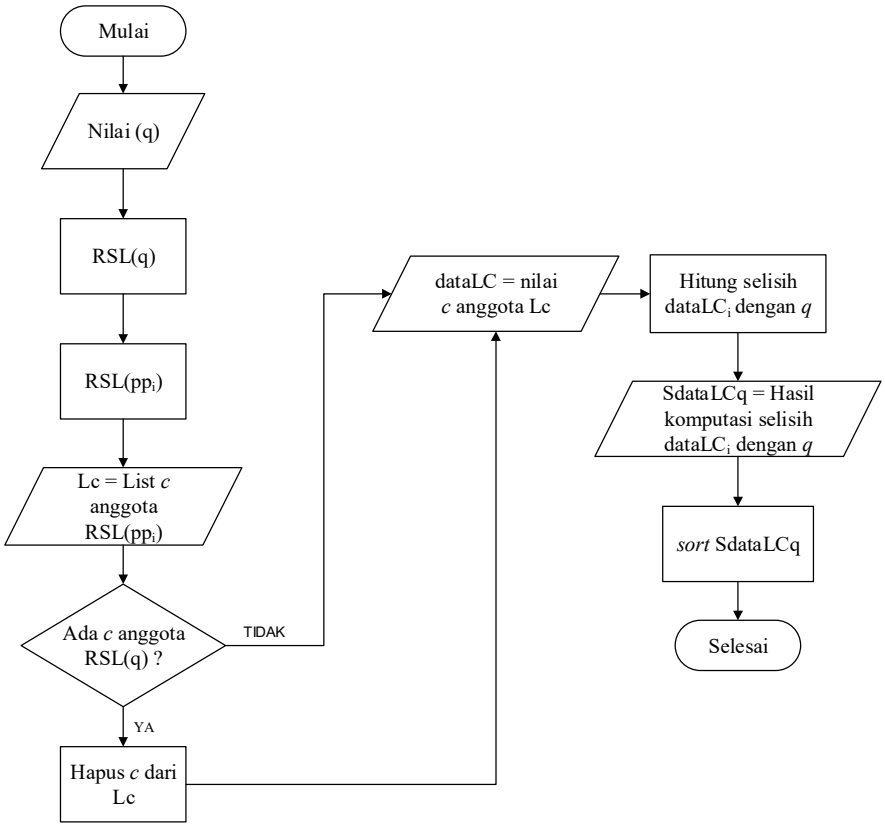
Proses modifikasi *query point* merupakan pendekatan *data refinement* yang diusulkan pada penelitian ini. Dengan melakukan modifikasi nilai q menjadi q' *output* yang diharapkan adalah munculnya q' sebagai anggota $K - MPP$.

Definisi 3. Modifikasi *query point* q dapat dilakukan dengan mempertimbangkan anggota RSL dari *promising product* $pp_i \in K - MPP$.

Untuk set dimensi data D dari tiap customer $c \in RSL_{(pp)}$ dan c bukan anggota $RSL_{(q)}$, nilai data minimal customer c pada dimensi d yang mendekati nilai q adalah Vd_{min}, Nilai q' dapat ditentukan dengan mengubah nilai pada dimensi ke-d pada *query point* q menjadi sama dengan nilai Vd_{min}.

Gambar 3.5 Alur Proses Modifikasi *Query*

Berdasarkan Definisi 3, dapat diketahui bahwa proses *data refinement* terdiri dari tiga tahap seperti yang ditunjukkan pada Gambar 3.5. Langkah awal yang perlu dilakukan sebelum proses modifikasi data adalah identifikasi RSL dari *query* q dan tiap $pp_i \in K - MPP$. Diagram alur keseluruhan pra-proses yang perlu dilakukan sebelum penentuan *data refinement* dapat dilihat pada Gambar 3.6.



Gambar 3.6 Diagram Alur Pra-Proses Modifikasi *Query*

Setelah mendapatkan Lc yang merupakan himpunan $C \in RSL(pp_i)$, kemudian akan dilakukan penghitungan selisih atau besar perbedaan nilai pada tiap dimensi d_i dari q terhadap nilai preferensi tiap *customer* c_i yang muncul pada RSL anggota $K - MPP$ yaitu pp_i. Hasil komputasi tersebut kemudian disimpan pada $SdataLCq$. Dikarenakan proses modifikasi *query* perlu

mempertimbangkan *cost* perubahan yang seminimal mungkin, lengkah pra-proses selanjutnya adalah *sorting* $SdataLCq$ berdasarkan nilai minimal dari keseluruhan dimensi data d.

Contoh 4. Berdasarkan *dataset* pada Tabel 2.1, nilai *preference why-not* $3-MPP$ produk p_8 adalah (16,6), sedangkan RSL dari tiap anggota $3-MPP$ dan nilai *preferencenya* adalah c_{10} (20,13), c_7 (12,2), c_5 (10,10), c_3 (6,16), c_8 (18,6), c_2 (4,10), c_1 (2,8), dan c_4 (8,18). Selisih p_8 terhadap seluruh anggota RSL $3-MPP$ dapat dilihat pada $SdataLCq$ Tabel 3.5.

Tabel 3.5. Hasil Komputasi $SdataLCq$ *Why-Not Point* terhadap Anggota *RSL*(pp_i)

Customer	**Selisih**		$\boldsymbol{d_{min}}$
	$\boldsymbol{d_1}$	$\boldsymbol{d_2}$	
c_8	2	0	0
c_1	14	2	2
c_2	12	4	4
c_5	6	4	4
c_7	4	4	4
c_{10}	4	7	4
c_4	8	12	8
c_3	10	10	10

Setelah memperoleh tabel $SdataLCq$ pada pra-proses, modifikasi *query* dilakukan dengan mengubah nilai *query* q pada salah satu dimensinya dengan mempertimbangkan nilai *data point* anggota RSL(pp_i) yang memiliki selisih minimal pada $SdataLCq$. Dengan melakukan modifikasi *query* berdasarkan nilai preferensi *customer* c_i yang muncul sebagai anggota RSL(pp_i), maka c_i tersebut diharapkan dapat muncul sebagai anggota RSL(q'). Modifikasi *query* akan mengakibatkan adanya perubahan nilai skor *probability*. Perubahan nilai

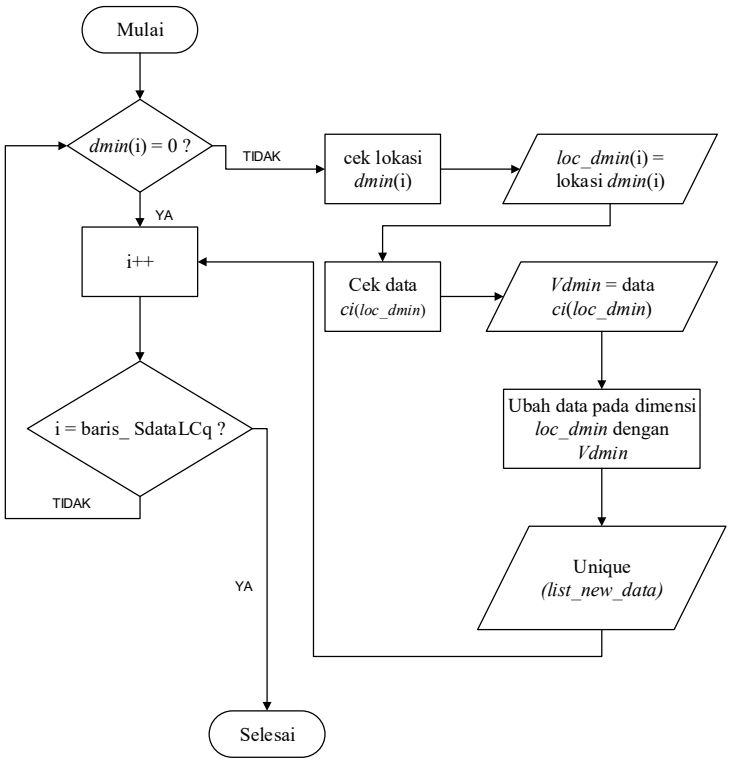
probability kemudian juga akan mengakibatkan berubahnya skor MC dan *ranking* $K - MPP$.

Diagram alur usulan algoritma *data refinement* pada penelitian ini tercantum pada Gambar 3.7. Apabila dalam suatu data terdapat dua atau lebih dimensi data yang merupakan anggota loc_dmin, maka jumlah *query* baru yang terbentuk dari data tersebut akan berjumlah sama dengan jumlah dimensi data dari data tersebut. Seluruh *query* baru dari hasil *data refinement* akan disimpan pada variabel $list\_new\_data$ yang merupakan himpunan kemungkinan perubahan *query* yang nilainya unik.

Contoh 5. Berdasarkan hasil tabel $SdataLCq$ pada Tabel 3.5, *data point* c_8 memiliki nilai terdekat dengan *why-not point* q pada dimensi d_2, namun karena c_8 telah menjadi anggota $RSL_{(p_8)}$, maka nilai Vd_{min} adalah 8 yang merupakan nilai *preference* c_1 pada dimensi d_2. Nilai q' kemudian dapat ditentukan dengan mengubah nilai d_2 dari q menjadi sama dengan nilai Vd_{min}, sehingga diperoleh *query* baru q' = 16,8. Proses ini akan terus dilakukan pada keseluruhan data pada tabel $SdataLCq$, sehingga diperoleh $list\_new\_data$ pada Tabel 3.6.

Tabel 3.6 Hasil Modifikasi *Query*

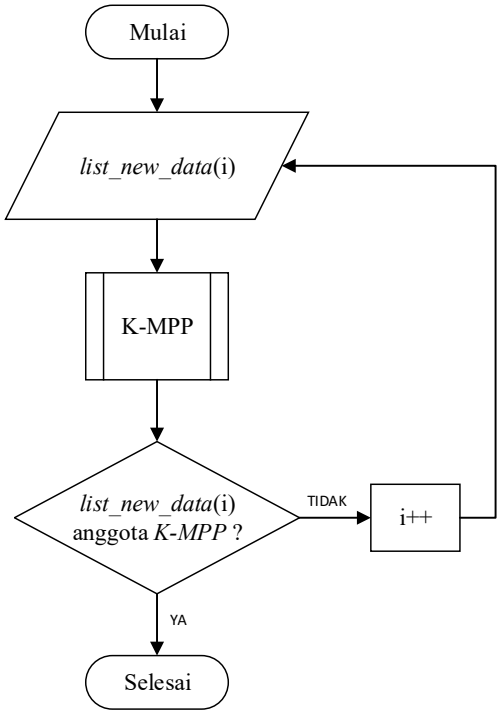
No	Hasil q'
1	16,8
2	16,10
3	16,2
4	12,6
5	20,6
6	8,6
7	16,16
8	6,6



Gambar 3.7. Diagram Alur Proses *Data Refinement*

Tidak semua anggota $list\_new\_data$ dapat menjadikan *why-not point* menjadi anggota $K - MPP$, sehingga pada tahap berikutnya diperlukan proses validasi. Tujuan dari proses validasi adalah untuk melakukan komputasi $K - MPP$ dengan nilai baru *why-not point* yang telah dimodifikasi menjadi salah satu nilai dari $list\_new\_data$. Diagram alur proses validasi terdapat pada Gambar 3.8.

Karena tabel $list\_new\_data$ telah berisi nilai modifikasi *query* berdasarkan perubahan data terkecil pada salah satu dimensinya, maka proses validasi akan dihentikan apabila telah diperoleh satu hasil q' dari $list\_new\_data$ yang merupakan anggota $K - MPP$. Berdasarkan tiga tahap modifikasi *query* yang telah dilakukan, akan diperoleh *data refinement* dengan nilai perubahan terkecil, sehingga *cost* perubahan yang diperlukan juga merupakan *cost* yang paling minimal.



Gambar 3.8. Diagram Alur Proses Validasi

3.3 Implementasi Algoritma

Implementasi dari metode yang diusulkan dilakukan pada perangkat lunak Matlab versi R2017a terhadap data sintetis dengan persebaran data independen (IND). Dengan menggunakan data sintetis independen, *dataset* produk dan *customer* akan berisi data yang nilainya *random*.

3.4 Pengujian

Pengujian dan evaluasi yang dilakukan pada penelitian ini dilakukan untuk mengetahui apakah pendekatan *data refinement* yang diusulkan dapat menjawab *why-not question* pada $K - MPP$. Pada bagian ini akan dibahas mengenai skenario pengujian yang akan dilakukan agar dapat tepat sasaran dan sesuai dengan tujuan penelitian. Selanjutnya, akan dilakukan proses evaluasi untuk mendapatkan nilai kuantitatif dari skenario uji coba yang dilakukan. Pada penelitian ini, metrik yang digunakan adalah jumlah $list\_new\_data$ yang diusulkan sebagai saran *data refinement*, keberhasilan pendekatan *data refinement* yang dievaluasi pada tahap validasi modifikasi *query* dan rata-rata waktu yang dibutuhkan untuk menemukan usulan *data refinement* dengan *cost*

minimal yang mampu membuat *why-not point* masuk sebagai anggota $K - MPP$. Hasil dari proses evaluasi ini diintepretasikan untuk menjawab apakah tujuan penelitian telah tercapai.

Terdapat beberapa variabel bebas yang akan dimodifikasi nilainya sebagai skenario pengujian, yaitu:

a. Kardinalitas data (jumlah data yang akan dievaluasi)
   Nilai : 5.000, 10.000, 20.000, 30.000, dan 50.000
   Default value: 20.000
b. Jumlah dimensi atau total atribut dari *dataset* yang digunakan
   Nilai : 2, 3, 5, 7, dan 10
   Default value: 3
c. ΔK (selisih nilai K pada $K - MPP$ dengan K' atau *ranking* dari *why-not point*)
   Nilai : 1, 3, 5, 7, dan 10
   Default value: 3
d. Tipe data yang digunakan
   Jenis : *dataset* sintetis (Independen dan *anti-correlated*) dan *dataset real* " Forest Cover Type Data Set" yang diperoleh dari halaman *website* https://archive.ics.uci.edu/ml/datasets/covertype

Evaluasi dilakukan dengan cara menjalankan keseluruhan tahap Algoritma terhadap variasi skenario yang ada dengan melakukan pengulangan sebanyak 20 kali evaluasi pada tiap skenario.

3.5 Analisa Hasil

Analisa digunakan untuk mengetahui capaian yang dihasilkan dari metode yang dirancang. Capaian ini dapat dilihat berdasarkan metrik hasil pengujian yang diperoleh pada beberapa skenario pengujian yang dilakukan. Pada metrik jumlah $list\_new\_data$, perlu diketahui apakah terdapat korelasi antara jumlah kombinasi *data refinement* yang diusulkan terhadap keseluruhan skenario yang diujikan. Selanjutnya, pada metrik kedua akan dianalisa apakah pendekatan *data refinement* berhasil menjawab *why-not* $K - MPP$

pada keseluruhan variasi skenario uji coba. Hasil pengujian rata-rata waktu juga akan dianalisa korelasinya terhadap skenario pengujian yang dilakukan.

3.6 Dokumentasi

Pada tahap dokumentasi akan dilakukan penulisan laporan hasil penelitian yang dilakukan. Tujuan dari tahap ini adalah menghasilkan dokumentasi tertulis dari seluruh tahap penelitian.

[Halaman ini sengaja dikosongkan]

BAB 4
HASIL DAN PEMBAHASAN

Pada bab ini akan dijelaskan mengenai proses implementasi metode yang diusulkan dan pembahasan berupa analisa dari hasil pengujian yang telah dilakukan terhadap skenario yang telah ditentukan pada penjelasan bab sebelumnya. Penjelasan mengenai proses implementasi meliputi lingkungan implementasi perangkat keras dan lunak yang digunakan pada penelitian ini dan penjabaran tahap implementasi berupa *pseudocode*. Bagian pembahasan akan menjelaskan mengenai lingkungan uji coba, data uji coba, skenario uji coba yang meliputi uji fungsionalitas dan uji performa, dan analisis hasil uji coba.

4.1 Lingkungan Implementasi Perangkat Lunak

Lingkungan implementasi algoritma pada penelitian ini meliputi perangkat keras dan perangkat lunak. Perangkat keras yang dimaksud adalah jenis *processor* dan *memory* (RAM) yang digunakan, sedangkan perangkat lunak adalah sistem operasi serta *tools* yang digunakan untuk implementasi. Rincian tersebut dapat dilihat pada Tabel 4.1 dibawah ini.

Tabel 4.1. Rincian Lingkungan Implementasi

Jenis Lingkungan	**Rincian**	
Perangkat Keras	*Processor*	Intel Core i5-7400 3.00 Ghz
	Memory (RAM)	8192 MB
Perangkat Lunak	Sistem Operasi	Windows 10 Pro 64-bit
	Tools	Matlab 2017a

4.2 Implementasi Algoritma

Hasil perancangan algoritma yang telah dibahas pada bab sebelumnya akan diolah menjadi *pseudocode* pada bab ini. *Pseudocode* ini akan diterapkan pada *tools* untuk pengujian, dalam hal ini adalah *Matlab*. Tahap implementasi algoritma pada penelitian ini secara garis besar terdiri dari empat tahap, yaitu: menentukan *why-not point*,

menentukan penyebab *why-not question* atau tidak munculnya *why-not point* pada hasil *query* $K - MPP$, modifikasi *query* dan validasi.

4.2.1 Implementasi Penentuan *Why-not Point*

```
1         Input whynotpoint
2
3         for i = 1:baris_sortedMC
4            if sortedMC(i) = whynotpoint
5               rankw = sortedMC(i)
6               new_KMPP = sortedMC(1:rankw)
7            end
8         end
9
10        Display(rankw)
11        Display(new_KMPP)
```

Gambar 4.1. *Pseudocode* Penentuan *Why-not Point*

Implementasi penentuan *why-not point* diawali dengan proses input *user feedback* yang merupakan data produk yang bukan merupakan anggota $K - MPP$ (baris 1 Gambar 4.1). Setelah diperoleh *why-not point*, kemudian akan dilakukan proses identifikasi posisi peringkat dari *why-not point* tersebut (baris 3 – 8).

4.2.2 Implementasi Penentuan *Penyebab Why-Not Question*

```
1         nRSL_w =  length(hasilRSL(w))
2
3         for i = 1:baris_hasilKMPP
4         nRSL_KMPP = length(hasilRSL(hasilKMPP(i)))
5         end
6
7         for j = 1:baris_nRSL_KMPP
8            if nRSL_KMPP(j) < nRSL_w
9               cek_nRSL = 1
10              break
11           else
12              cek_nRSL = 0
13           end
14        end
15
16        if cek_nRSL == 1
17           sorted_hasilProb = sort(hasilProb,'descend')
18           MVC = sorted_hasilProb(1:k)
19           Display ("MVC")
20        else
21           Display ("nRSL")
22        end
```

Gambar 4.2 *Pseudocode* Penentuan Penyebab *Why-not Question*

Setelah diperoleh *why-not point* pada proses sebelumnya, kemudian akan dilakukan proses identifikasi penyebab tidak munculnya *why-not point* pada anggota $K - MPP$ yang alurnya dapat dilihat pada Gambar 4.2. Pada baris pertama dilakukan identifikasi jumlah RSL dari *why-not point*. Baris 3 – 5 merupakan proses identifikasi jumlah RSL dari anggota $K - MPP$. Setelah mengetahui jumlah RSL, pada baris 7- 14 kemudian akan dilakukan evaluasi apakah terdapat jumlah RSL anggota $K - MPP$ yang nilainya kurang dari jumlah RSL dari *why-not point*. Baris 16 – 22 kemudian akan menjelaskan penyebab dari *why-not question* yang muncul pada hasil *query* $K - MPP$. Apabila tidak terdapat jumlah RSL anggota $K - MPP$ yang lebih sedikit daripada jumlah RSL *why-not point* (cek_nRSL = 0) maka penyebab *why-not point* bukan merupakan anggota $K - MPP$ adalah kurangnya jumlah RSL, dan sebaliknya apabila cek_nRSL = 1 maka penyebabnya adalah tidak munculnya MVC pada anggota RSL *why-not point*.

4.2.3 Implementasi Modifikasi *Query*

```
1         dataw = datapq(w)
2         wnRSL = hasilRSL(w)
3
4         for i = 1:baris_hasilRSL
5            a_KMPP = hasilKMPP(i)
6            KMPP_RSL = [hasilRSL(a_KMPP)]
7         end
8
9         for j = 1:baris_KMPP_RSL
10           Lc = find(unique(KMPP_RSL)) ∉ wnRSL
11        end
12
13        for k = 1:baris_u_KMPPRSL
13           data_Lc = datac(u_KMPPRSL(k))
15        end
16
17        for l = 1:baris_data_u_KMPPRSL
18           SdataLCq = dataw - data_u_KMPPRSL(l)
19        end
20
21        sort(SdataLCq, 'ascend')
22
23        u_selisih = find(unique(selisih)) ≠ 0
24        for m = 1:baris_u_selisih
25           dmin = min(u_selisih)
26           for n = 1:baris_selisih
27               for o = 1:d_selisih
28                   if selisih(n,m) = min_selisih
29                       loc_dmin = [(n,m)]
30                       vdmin = [datac(loc_dmin)]
```

```
31                      min_d = [indeks_datac(n) loc_dmin vdmin]
32                  end
33               end
34           end
35           if nRSL ≠ 0
36              cekKMVC = [sort_prob <= v]
37                   for p = 1:baris_min_d
38                       if ismember((min_d(p)),cekKMVC) ≠ 1
39                           min_d_MVC = [datac(cekMVC) selisih_min_
                         datac(cekMVC)]
40                       end
41                       hasil_min_d = min_d_MVC
42                  end
43           else
44              hasil_min_d = min_d
45           end
46           for q = 1:baris_hasil_min_d
47           list_new_data ← hasil ubah dataw pada dimensi loc_dmin
             dengan hasil_min_d[2]
48           end

49           do validasi
50        end
```

Gambar 4.3. *Pseudocode* Implementasi Modifikasi *Query*

Pada proses modifikasi *query* pertama-tama akan dilakukan identifikasi RSL dari *why-not point* dan anggota $K - MPP$ (baris 1- 7). Setelah mendapatkan Lc dan $data\_Lc$ yang merupakan himpunan anggota RSL dari anggota $K - MPP$ dan nilai *preference* dari *c* yang muncul pada Lc, kemudian akan dilakukan penghitungan selisih atau besar perbedaan nilai pada tiap dimensi data dari *why-not point* q terhadap nilai preferensi tiap *customer*. Hasil komputasi tersebut kemudian disimpan pada $SdataLCq$. Untuk menjamin *cost* perubahan yang seminimal mungkin, langkah selanjutnya adalah sorting $SdataLCq$ berdasarkan nilai minimal dari keseluruhan dimensi data nya (baris 9 – 21). Pada baris 23 – 48 dilakukan penentuan lokasi perubahan data, penentuan penyebab *why-not point* tidak muncul sebagai anggota $K - MPP$, dan proses penentuan *data refinement* yang diusulkan. Setelah memperoleh kumpulan variasi *data refinement* yang diusulkan pada variabel *list_new_data*, kemudian pada baris ke-49 akan dilakukan proses validasi yang akan dijelaskan pada sub bagian berikutnya.

Apabila pada proses validasi seluruh kombinasi *data refinement* pertama pada *list_new_data* gagal menjadi anggota $K - MPP$ maka akan dilakukan iterasi selanjutnya pada baris 24 - 48 hingga proses validasi berhasil menemukan *data refinement* yang dapat membuat *why-not point* masuk menjadi anggota $K - MPP$.

4.2.4 Implementasi Validasi

```
1         for i = 1:baris_list_new_data
2           do K-MPP
3           if list_new_data(i) ∉ hasil_KMPP
4               i++
5           else
6               display(list_new_data(i))
7                break;
8           end
9         end
```

Gambar 4.4. *Pseudocode* Validasi Data

Proses validasi data adalah proses pengecekan apakah $list\_new\_data(i)$ atau *data refinement* yang diusulkan telah dapat menjawab permasalahan *why-not* yang muncul. Alur proses validasi yang ditunjukkan pada Gambar 4.4. Dengan melakukan pengecekan satu persatu *list_new_data* pada fungsi $K - MPP$, apabila telah diperoleh *data refinement* yang dapat menyebabkan *why-not point* masuk menjadi anggota K-MPP maka proses validasi akan dihentikan.

4.3 Lingkungan Pengujian Algoritma

Lingkungan uji coba menjelaskan mengenai lingkungan yang digunakan dalam menguji implementasi pendekatan *data refinement* dalam mengatasi permasalahan *why-not question* pada hasil *query* $K - MPP$. Uji coba dilakukan pada perangkat dengan spesifikasi seperti yang tertera pada Tabel 4.2.

Tabel 4.2. Lingkungan Uji Coba

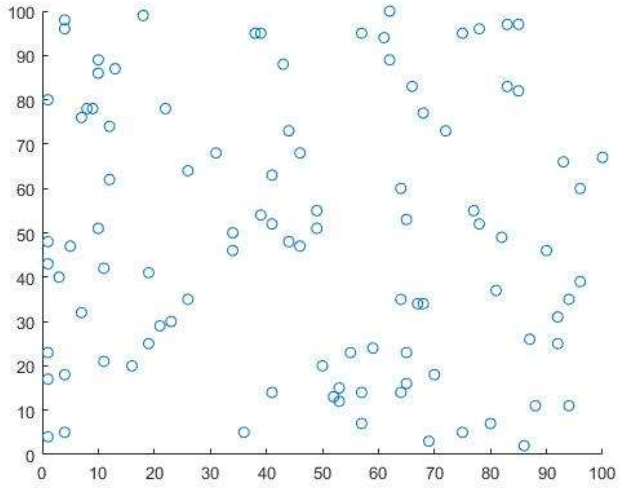
Jenis Lingkungan	**Rincian**	
Perangkat Keras	*Processor*	Intel Core i5-7400 3.00 Ghz
	Memory (RAM)	8192 MB
Perangkat Lunak	Sistem Operasi	Windows 10 Pro 64-bit
	Tools	Matlab 2017a

4.4 Data Pengujian

Tahap pengujian pada penelitian ini menggunakan 3 jenis data, yaitu data independen (IND), data *anti-corelated* (ANT) dan data *forest cover type* (FC),. Masing-masing jenis data memiliki variasi pada jumlah data n dan jumlah dimensi data d.

4.4.1 Data Independen (IND)

Data independen (IND) adalah himpunan data sintetis yang memiliki persebaran nilai atribut yang acak dan tidak saling terpengaruh antara atribut satu dengan lainnya maupun antar nilai suatu data dengan nilai data lainnya. Penggunaan data ini bertujuan untuk menguji performa algoritma jika berhadapan dengan data yang nilai atributnya tidak memiliki keterkaitan satu sama lain. Rentang nilai yang digunakan untuk setiap atribut adalah 1 – 100. Ilustrasi persebaran data pada tipe data independen dapat dilihat pada Gambar 4.5.

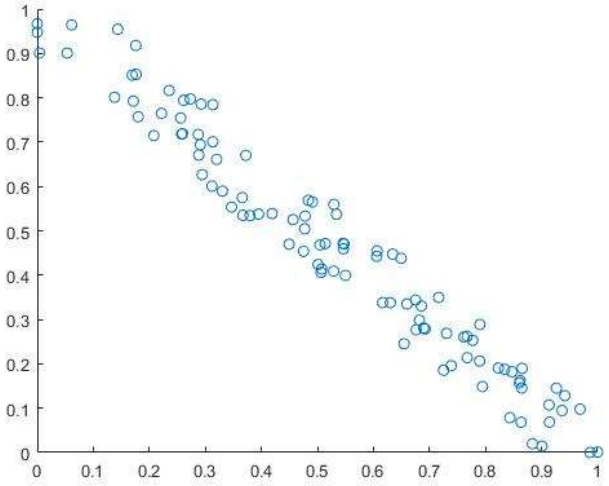


Gambar 4.5. Persebaran Data Pada Tipe Data Independen

4.4.2 Data *Anti-correlated* (ANT)

Data *anti-correlated* (ANT) adalah himpunan data sintetis yang memiliki persebaran nilai atribut yang saling bertolak belakang antara satu antribut dengan atribut lainnya, yang artinya sebuah data memiliki nilai yang sangat baik pada salah satu atributnya namun sangat buruk pada atribut lainnya.

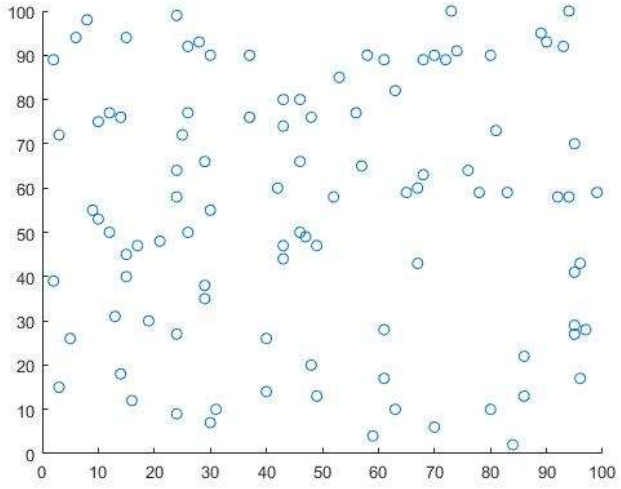
Penggunaan data ini bertujuan untuk menguji performa algoritma jika berhadapan dengan data yang nilai atributnya saling bertolak belakang dan memiliki relasi dominasi antar data paling rendah dibandingkan dengan jenis data lainnya. Rentang nilai yang digunakan untuk setiap atribut adalah 1 – 100. Ilustrasi persebaran data pada tipe data *anti-correlated* dapat dilihat pada Gambar 4.6.



Gambar 4.6. Persebaran Data Pada Tipe Data *Anti-correlated*

4.4.3 Data *Forest Cover type* (FC)

Data *Forest Cover Type* (FC) adalah himpunan data *real* yang diambil dari sumber sebenarnya. Penggunaan data ini bertujuan untuk menguji performa algoritma pada data dengan persebaran dan rentang nilai atribut yang bersifat saling berkaitan. Ilustrasi persebaran data pada *dataset forest cover type* dapat dilihat pada Gambar 4.7.



Gambar 4.7. Persebaran Data Pada Tipe Data *Forest Cover Type*

4.5 Skenario Pengujian

Pengujian dilakukan kepada setiap tipe *dataset* (independen, *anti-correlated*, dan *forest cover type*) dengan berbagai variasi nilai untuk setiap variabel bebas yang ada (kardinalitas, d, dan ΔK). Adapun skenario uji coba yang dilakukan sebagai berikut:

a. Variasi kardinalitas data: 5.000, 10.000, 20.000, 30.000, dan 50.000.

b. Variasi jumlah dimensi (d): 2, 3, 5, 7, dan 10.

c. Variasi ΔK atau selisih *ranking why-not point* dengan K pada $K - MPP$: 1, 3, 5, 7, dan 10.

Selain nilai variabel bebas diatas, terdapat pula nilai tetap yang merupakan *default value* dan tidak dilakukan perubahan pada setiap uji coba terhadap variabel bebas tertentu, yaitu:

a. Kardinalitas data: 20.000.

b. Jumlah dimensi: 3.

c. ΔK: 3.

Sebagai contoh, ketika dilakukan proses pengujian pada skenario pertama, dimana variabel kardinalitas data akan menjadi variabel bebas yang nilainya bervariasi sesuai dengan skenario yang telah ditentukan (5.000, 10.000, 20.000, 30.000, dan 50.000), variabel bebas yang lain akan menggunakan nilai tetap *default value* d = 3, dan ΔK: 3.

Skenario pengujian berfokus pada seberapa baik program dapat memberikan solusi atas metode yang diusulkan. Secara garis besar keberhasilan metode usulan akan dibandingkan dengan durasi waktu eksekusi yang dibutuhkan.

Adapun detail skenario pengujian yang dilakukan sebagai berikut.

4.5.1 Skenario Variasi Kardinalitas Data

Variasi kardinalitas data (n) yang akan dievaluasi adalah dengan jumlah data sebesar 5.000, 10.000, 20.000, 30.000, dan 50.000. Detail skenario dapat dilihat pada Tabel 4.3.

Tabel 4.3. Skenario Variasi Kardinalitas Data

Skenario	Jumlah Dimensi (d)	Kardinalitas data (n)	ΔK
A01	3	5.000	3
A02	3	10.000	3
A03	3	20.000	3
A04	3	30.000	3
A05	3	50.000	3

4.5.2 Skenario Variasi Jumlah Dimensi Data

Variasi jumlah dimensi data (d) yang akan dievaluasi adalah dengan jumlah dimensi sebesar 2, 3, 5, 7, dan 10. Detail skenario dapat dilihat pada Tabel 4.4.

Tabel 4.4. Skenario Variasi Jumlah Dimensi Data

Skenario	Jumlah Dimensi (d)	Kardinalitas data (n)	ΔK
B01	2	20.000	3
B02	3	20.000	3
B03	5	20.000	3
B04	7	20.000	3
B05	10	20.000	3

4.5.3 Skenario Variasi ΔK

Variasi jumlah selisih *ranking why-not point* terhadap K pada $K - MPP$ (ΔK) yang akan dievaluasi adalah dengan jumlah selisih sebesar 1, 3, 5, 7, dan 10. Detail skenario dapat dilihat pada Tabel 4.5.

Tabel 4.5. Skenario Variasi ΔK

Skenario	Jumlah Dimensi (d)	Kardinalitas data (n)	ΔK
C01	3	20.000	1
C02	3	20.000	3
C03	3	20.000	5
C04	3	20.000	7
C05	4	20.000	10

4.6 Hasil Pengujian

Berdasarkan hasil pengujian yang telah dilakukan, dalam penelitian ini terdapat tiga metrik yang akan dianalisa. Metrik pertama adalah jumlah $list\_new\_data$ yang diusulkan sebagai saran *data refinement*, metrik kedua adalah rata-rata waktu yang dibutuhkan untuk menemukan usulan *data refinement*, dan metrik ketiga adalah rata-rata waktu yang dibutuhkan untuk menemukan *data refinement* yang mampu menyelesaikan permasalahan *why-not question* pada $K - MPP$. Penentuan *data refinement* yang berhasil menjadi anggota $K - MPP$ dilakukan pada tahap validasi modifikasi *query*. Hasil pengujian akan disajikan dalam tampilan tabel hasil dan grafik selama proses pengujian dilakukan. Setiap pengujian akan dilakukan pada beberapa *dataset* yang sudah dijelaskan pada sub bab sebelumnya (IND = independen, ANT = *anti-correlated*, dan FC = *forest cover type*)

4.6.1 Hasil Pengujian Variasi Kardinalitas Data

Dalam pengujian variasi jumlah data pada data IND,ANT, dan FC, diperoleh hasil pengujian berupa jumlah variasi *data refinement*, Δt (rata-rata waktu yang dibutuhkan untuk proses penentuan variasi data), dan rata-rata waktu untuk menjalankan proses validasi dari metode yang diusulkan pada Tabel 4.6, Tabel 4.7

dan Tabel 4.8. Hasil dari tiap pengujian variasi kardinalitas data merupakan rata-rata hasil dari 20 kali proses evaluasi yang telah dilakukan.

Tabel 4.6. Hasil Pengujian Variasi Kardinalitas Data Pada *Dataset* Independen

Jenis Data	*d*	Jumlah Data	ΔK	Jumlah Variasi	Δt (s)	Validasi (s)
IND	3	5000	3	17	1.56	340
		10000		39	1.61	675
		20000		115	1.77	2100
		30000		139	1.84	2798
		50000		153	1.88	3230

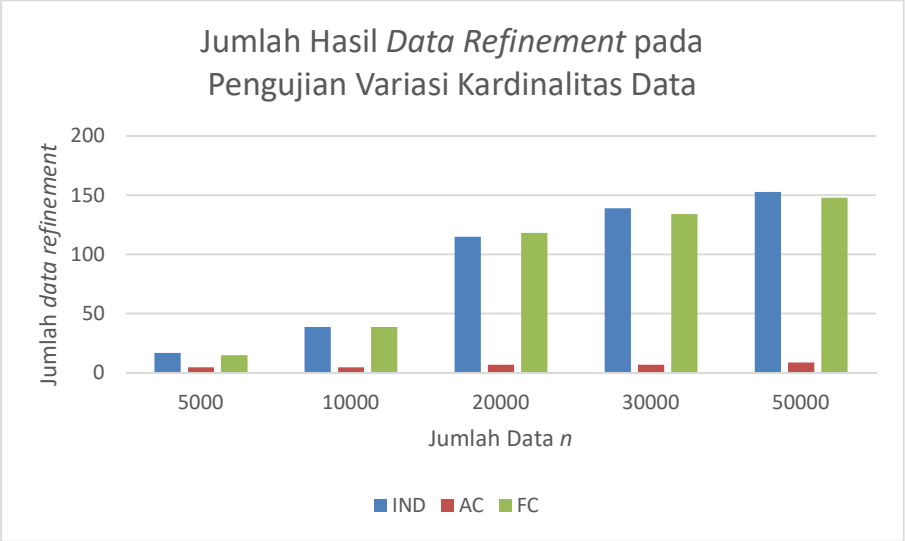
Tabel 4.7. Hasil Pengujian Variasi Kardinalitas Data Pada *Dataset Anti-correlated*

Jenis Data	*d*	Jumlah Data	ΔK	Jumlah Variasi	Δt (s)	Validasi (s)
ANT	3	5000	3	5	1.13	25
		10000		5	1.17	47
		20000		7	1.28	210
		30000		7	1.24	176
		50000		9	1.33	328

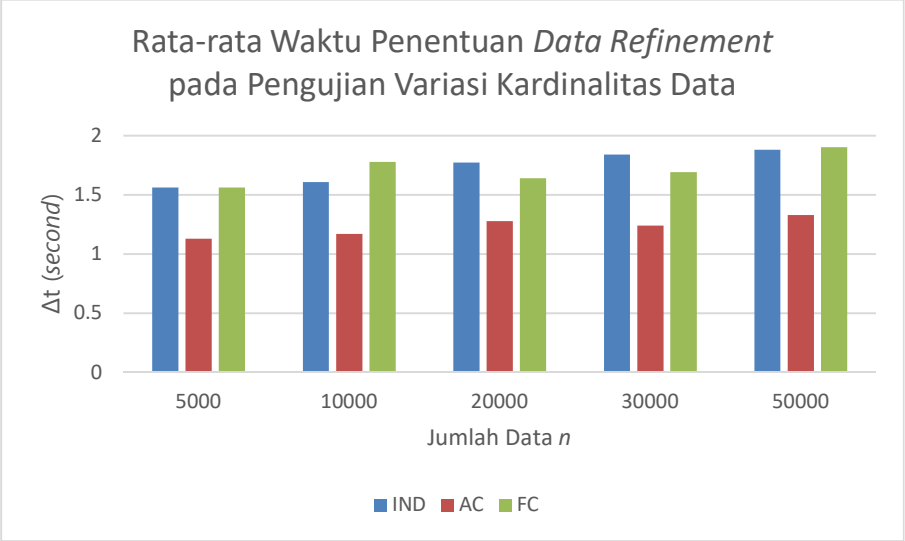
Tabel 4.8. Hasil Pengujian Variasi Kardinalitas Data Pada *Dataset Forest Cover Type*

Jenis Data	*d*	Jumlah Data	ΔK	Jumlah Variasi	Δt (s)	Validasi (s)
FC	3	5000	3	15	1.56	322
		10000		39	1.78	710
		20000		118	1.64	1988
		30000		134	1.69	2679
		50000		148	1.9	3123

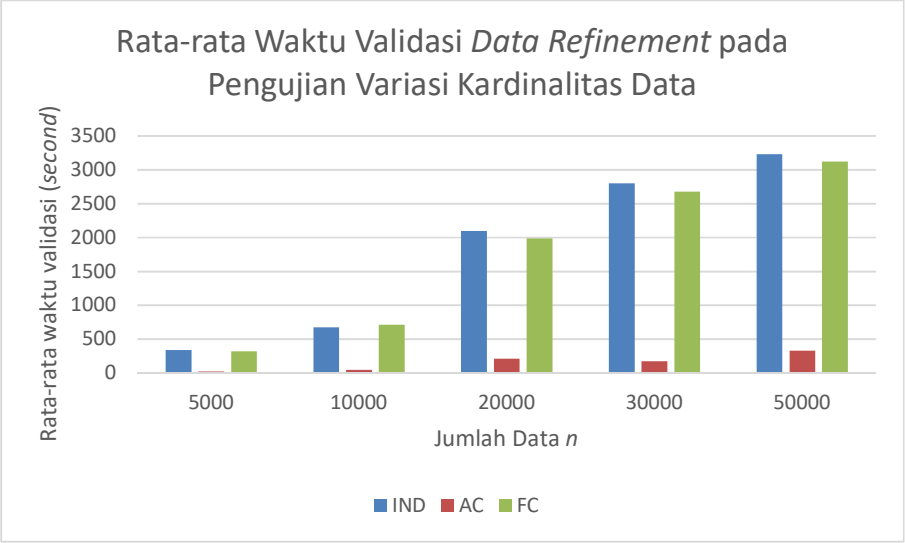
Dapat dilihat pada Gambar 4.8, Gambar 4.9 dan Gambar 4.10 bahwa jumlah variasi *data refinement* yang dihasilkan pada tiap variasi jumlah data akan mengalami peningkatan pada jumlah data yang semakin banyak. Begitu pula dengan waktu validasi yang dibutuhkan untuk melakukan pengecekan akhir apakah variasi *data refinement* yang dibentuk telah dapat mengatasi permasalahan *why-not* $K - MPP$ yang muncul.



Gambar 4.8. Hasil Pengujian Variasi Kardinalitas Data pada *Dataset* Independen



Gambar 4.9. Hasil Pengujian Variasi Kardinalitas Data pada *Dataset Anti-correlated*



Gambar 4.10. Hasil Pengujian Variasi Kardinalitas Data pada *Dataset Forest Cover Type*

Selain itu, pada Gambar 4.8 dan Gambar 4.10 dapat disimpulkan bahwa jumlah variasi *data refinement* yang dihasilkan pada tipe data IND dan FC cenderung tidak jauh berbeda, sedangkan pada data ANT pada Gambar 4.9 memiliki jumlah variasi yang jauh berbeda dan dalam perubahannya jumlah *data refinement* yang dihasilkan relatif lebih konstan dibandingkan dengan kedua tipe data lainnya. Hal ini dikarenakan oleh karakteristik persebaran data yang dimiliki oleh tiap jenis data. Pada data IND dan FC, data cenderung lebih tersebar pada tiap dimensi data yang dimiliki, dibandingkan dengan dengan data ANT.

Pada keseluruhan hasil pengujian yang telah dilakukan, dapat pula dilihat bahwa waktu yang dibutuhkan untuk menentukan *data refinement* relatif konstan dan juga akan mengalami sedikit peningkatan pada jumlah data yang lebih besar. Pada ketiga tabel diatas, diperoleh nilai minimal sebesar 1.22 s dan maksimal sebesar 1.9 s.

4.6.2 Hasil Pengujian Variasi Dimensi Data

Dalam pengujian variasi dimensi data pada data IND,ANT, dan FC, diperoleh hasil pengujian berupa jumlah variasi *data refinement*, Δt (rata-rata waktu yang

dibutuhkan untuk proses penentuan variasi data), dan rata-rata waktu untuk menjalankan proses validasi dari metode yang diusulkan pada Tabel 4.9, Tabel 4.10 dan Tabel 4.11. Hasil dari tiap pengujian variasi jumlah dimensi data merupakan rata-rata hasil dari 20 kali proses evaluasi yang telah dilakukan.

Tabel 4.9. Hasil Pengujian Variasi Jumlah Dimensi Data Pada *Dataset* Independen

Jenis Data	*d*	Jumlah Data	ΔK	Jumlah Variasi	Δt (s)	Validasi (s)
IND	2	20000	3	172	2.94	3782
	3			112	1.56	1986
	5			98	1.32	1876
	7			87	1.32	1479
	10			71	1.33	1283

Tabel 4.10. Hasil Pengujian Variasi Jumlah Dimensi Data Pada *Dataset Anti-correlated*

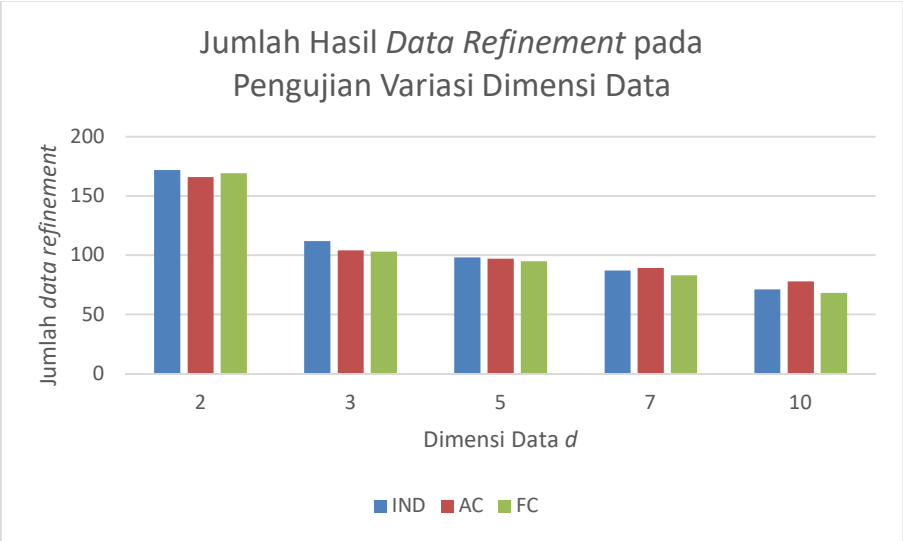
Jenis Data	*d*	Jumlah Data	ΔK	Jumlah Variasi	Δt (s)	Validasi (s)
ANT	2	20000	3	166	1.87	3457
	3			104	1.28	1894
	5			97	1.27	998
	7			89	1.22	831
	10			78	1.2	819

Tabel 4.11. Hasil Pengujian Variasi Jumlah Dimensi Data Pada *Dataset Forest Cover Type*

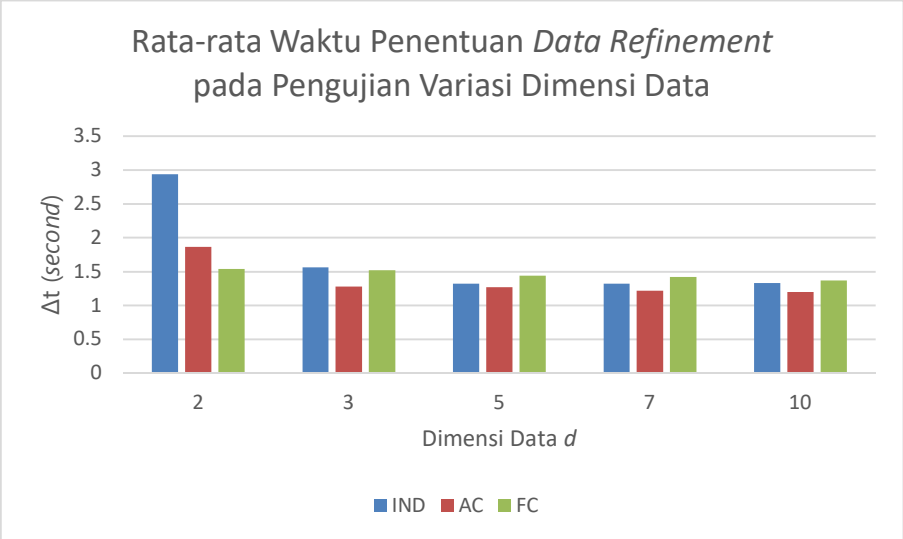
Jenis Data	*d*	Jumlah Data	ΔK	Jumlah Variasi	Δt (s)	Validasi (s)
FC	2	20000	3	169	1.54	3566
	3			103	1.52	1992
	5			95	1.44	1772
	7			83	1.42	1362
	10			68	1.37	1003

Dapat dilihat pada Gambar 4.11, Gambar 4.12 dan Gambar 4.13 bahwa jumlah variasi *data refinement* yang dihasilkan akan semakin sedikit pada jumlah dimensi data yang semakin banyak. Begitu pula dengan waktu validasi yang dibutuhkan

untuk melakukan pengecekan akhir apakah variasi *data refinement* yang dibentuk telah dapat mengatasi permasalahan *why-not* $K - MPP$ yang muncul.



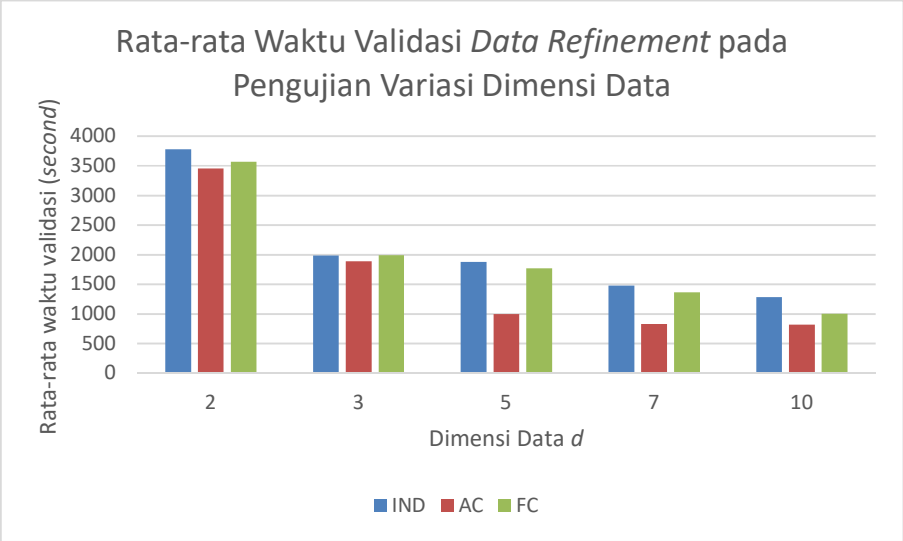
Gambar 4.11. Hasil Pengujian Variasi Jumlah Dimensi Data pada *Dataset* Independen



Gambar 4.12. Hasil Pengujian Variasi Jumlah Dimensi Data pada *Dataset Anti-correlated*

Berbeda dari skenario pengujian sebelumnya, pada skenario variasi jumlah dimensi data dapat disimpulkan bahwa pada ketiga tipe data jumlah variasi *data*

refinement yang dihasilkan cenderung tidak jauh berbeda. Hal ini disebabkan oleh jumlah hasil komputasi *skyline* yang sama-sama akan memiliki hasil yang lebih sedikit pada tiap peningkatan dimensi data. Selain itu, waktu penentuan variasi *data refinement* pada penelitian ini juga akan mengalami penurunan dikarenakan jumlah variasi *data refinement* juga akan semakin sedikit pada jumlah dimensi data yang semakin tinggi.



Gambar 4.13. Hasil Pengujian Variasi Jumlah Dimensi Data pada *Dataset Forest Cover Type*

4.6.3 Hasil Pengujian Variasi ΔK

Dalam pengujian variasi ΔK pada data IND,ANT, dan FC, diperoleh hasil pengujian berupa jumlah variasi *data refinement*, Δt (rata-rata waktu yang dibutuhkan untuk proses penentuan variasi data), dan rata-rata waktu untuk menjalankan proses validasi dari metode yang diusulkan pada Tabel 4.12, Tabel 4.13 dan Tabel 4.14. Hasil dari tiap pengujian variasi ΔK merupakan rata-rata hasil dari 20 kali proses evaluasi yang telah dilakukan.

Sama dengan hasil pada skenario pertama, pada skenario variasi ΔK dapat dilihat pada Gambar 4.14, Gambar 4.15 dan Gambar 4.16 bahwa jumlah variasi *data refinement* yang dihasilkan pada tiap variasi jumlah data akan mengalami

peningkatan pada jumlah ΔK yang semakin tinggi. Begitu pula dengan waktu validasi yang dibutuhkan untuk melakukan pengecekan akhir apakah variasi *data refinement* yang dibentuk telah dapat mengatasi permasalahan *why-not* $K - MPP$ yang muncul.

Tabel 4.12. Hasil Pengujian Variasi ΔK pada *Dataset* Independen

Jenis Data	*d*	Jumlah Data	ΔK	Jumlah Variasi	Δt (s)	Validasi (s)
IND	3	20000	1	57	2.31	892
			3	113	2.99	1562
			5	268	3.11	3862
			7	563	3.43	6610
			10	665	3.48	9234

Tabel 4.13 Hasil Pengujian Variasi ΔK Pada *Dataset Anti-correlated*

Jenis Data	*d*	Jumlah Data	ΔK	Jumlah Variasi	Δt (s)	Validasi (s)
ANT	3	20000	1	5	1.34	15
			3	7	1.41	11
			5	11	1.55	25
			7	21	1.62	38
			10	33	1.69	49

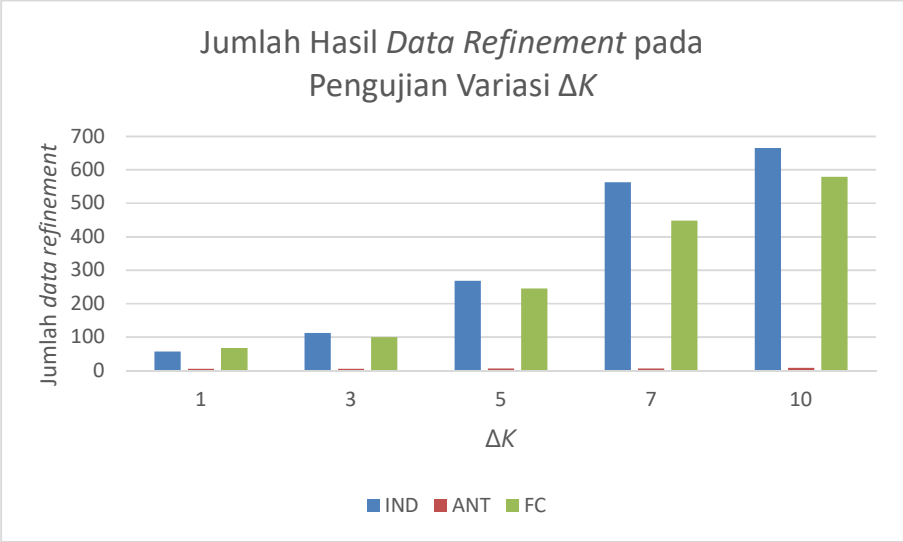
Tabel 4.14 Hasil Pengujian Variasi ΔK Pada *Dataset Forest Cover Type*

Jenis Data	*d*	Jumlah Data	ΔK	Jumlah Variasi	Δt (s)	Validasi (s)
FC	3	20000	1	68	2.64	1325
			3	100	2.71	1897
			5	246	2.98	3348
			7	448	3.21	4779
			10	579	3.28	5691

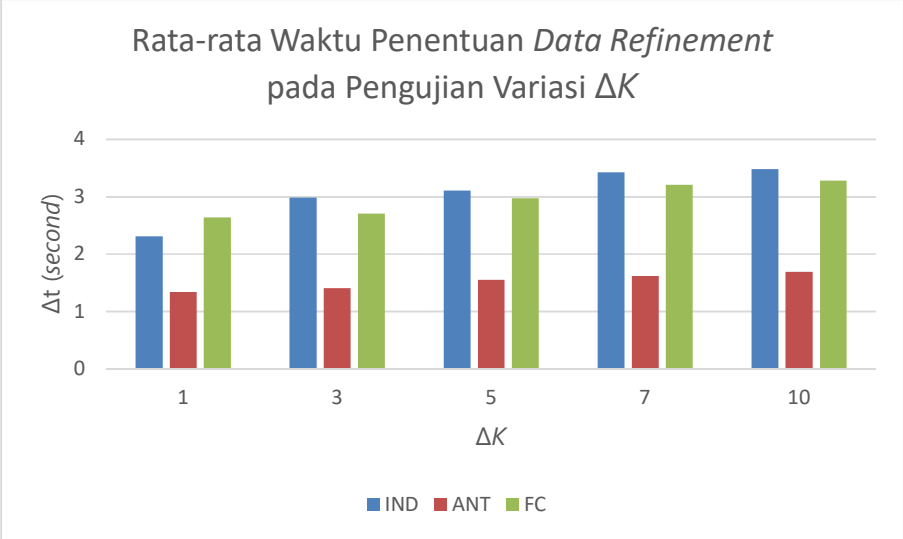
Jumlah variasi *data refinement* yang dihasilkan pada tipe data IND dan FC juga cenderung tidak jauh berbeda, sedangkan pada data ANT yang ditunjukkan pada Gambar 4.15 memiliki jumlah variasi *data refinement* jauh berbeda dan perubahan

jumlahnya relatif lebih konstan dibandingkan dengan kedua tipe data lainnya. Hal ini dikarenakan oleh karakteristik persebaran data IND dan FC yang data cenderung lebih tersebar pada tiap dimensi data yang dimiliki, dibandingkan dengan dengan data ANT.

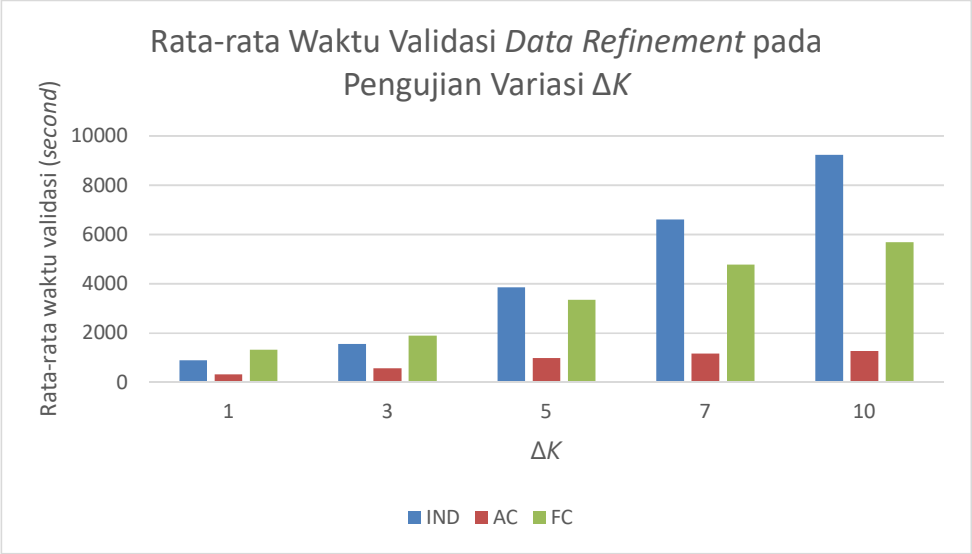
Pada keseluruhan hasil pengujian yang telah dilakukan, dapat pula dilihat bahwa waktu yang dibutuhkan untuk menentukan *data refinement* relatif konstan dan juga akan mengalami sedikit peningkatan pada Δ*K* yang semakin besar.



Gambar 4.14. Hasil Pengujian Variasi Δ*K* pada *Dataset* Independen



Gambar 4.15. Hasil Pengujian Variasi Δ*K* Pada *Dataset Anti-correlated*



Gambar 4.16. Hasil Pengujian Variasi Δ*K* pada *Dataset Forest Cover Type*

[Halaman ini sengaja dikosongkan]

BAB 5
KESIMPULAN DAN SARAN

Pada bab ini akan disampaikan kesimpulan yang diperoleh dari penelitian yang telah dilakukan dan saran tentang pengembangan dari penelitian ini yang dapat dilakukan di masa yang akan datang.

5.1 Kesimpulan

Dari proses perencangan, implementasi dan pengujian, dapat diambil beberapa kesimpulan sebagai berikut:

1. Pada komputasi $K - MPP$ apabila muncul *why-not question* pada hasil *query* yang diberikan, langkah identifikasi penyebab *why-not question* dapat dilakukan dengan mengevaluasi anggota RSL dari *why-not point* atau produk yang bukan merupakan anggota $K - MPP$ dan RSL dari anggota $K - MPP$. Karena semakin tinggi nilai *market contribution* suatu produk, maka semakin besar pula peluang produk tersebut untuk muncul sebagai hasil $K - MPP$. Nilai *market contribution* diperoleh dari penjumlahan nilai *probability* dari tiap anggota RSL. Berdasarkan jumlah RSL dari *why-not point* dan anggota $K - MPP$, apabila jumlah RSL dari *why-not point* kurang dari jumlah RSL minimal dari anggota $K - MPP$ maka penyebab produk tersebut tidak menjadi anggota $K - MPP$ adalah kurangnya jumlah anggota RSL suatu produk. Sebaliknya, apabila jumlah RSL dari *why-not point* sama dengan atau lebih dari jumlah RSL minimal dari anggota $K - MPP$, maka penyebabnya adalah tidak munculnya *Most Valuable Customer* (MVC) pada anggota RSL.
2. Proses modifikasi *query point* merupakan pendekatan *data refinement* yang diusulkan pada penelitian ini. Dengan melakukan modifikasi nilai q menjadi q' *output* yang diharapkan adalah munculnya q' sebagai anggota K-MPP. Modifikasi *query point* q dapat dilakukan dengan mempertimbangkan anggota RSL dari *promising product* pp_i anggota K-MPP dan c bukan merupakan anggota RSL_q. Nilai data minimal *customer* c pada *dimensi* d yang mendekati nilai q adalah nilai q' baru yang akan memiliki nilai *cost* modifikasi yang paling kecil.

3. Dalam melakukan pengujian pendekatan *data refinement* pada tiga skenario yang berbeda (variasi kardinalitas data, variasi dimensi data, dan variasi ΔK), diperoleh hasil :
    a. Waktu yang dibutuhkan untuk mencari variasi *data refinement* cenderung konstan dan akan mengalami peningkatan pada kardinalitas data dan ΔK yang tinggi. Waktu terbaik dari hasil evaluasi variasi kardinalitas data diperoleh pada jumlah data 5.000 pada jenis data *anti-correlated*, yaitu sebesar 1.13 s. Sebaliknya, pada jumlah data 50.000 dengan jenis data *forest cover type* diperoleh waktu komputasi *data refinement* terlama yaitu sebesar 1.91 s. Pada evaluasi variasi ΔK, waktu terbaik diperoleh pada jenis data *anti-correlated* dengan $\Delta K = 1$, yaitu sebesar 1.34 s, sedangkan waktu terlama sebesar 3.48 s diperoleh pada $\Delta K = 10$ pada jenis data independen.
    b. Pada evaluasi variasi jumlah dimensi data, waktu yang dibutuhkan untuk mencari *data refinement* akan semakin cepat dikarenakan jumlah variasi *data refinement* yang dihasilkan juga akan semakin sedikit pada jumlah dimensi data yang tinggi.
    c. Proses validasi masih membutuhkan waktu yang cukup lama (dengan rentang antara 13 menit hingga 1 jam) dan nilainya akan semakin tinggi pada kardinalitas data dan ΔK yang besar, namun akan semakin cepat pada jumlah dimensi data yang tinggi.

5.2 Saran

Berikut beberapa saran yang diberikan untuk pengembangan penelitian ini lebih lanjut:

1. Untuk mengurangi waktu yang dibutuhkan pada proses validasi dibutuhkan algoritma yang mampu mengurangi area evaluasi yang tidak diperlukan. Hal tersebut dapat dilakukan dengan melakukan cek dominasi *data refinement* terhadap data lain pada *dataset*.
2. Penelitian dapat dikembangkan dengan menambahkan kondisi data terdistribusi atau data yang tidak pasti.

DAFTAR PUSTAKA

Bartolini, I., Ciaccia, P. & Patella, M., 2006. Salsa: computing the skyline without scanning the whole sky. *CIKM,* pp. 405-414.

Borzsony, S., Kossmann, D. & Stocker, K., 2001. The Skyline Operator. In: *Data Engineering. Proceedings. 17th International Conference.* s.l.:IEEE, pp. 421-430.

Dellis, E. & Seeger, B., 2007. Efficient computation of reverse skyline queries. *The VLDB Journal,* pp. 291-302.

Islam, M. S., 2013. *On Answering Why and Why-not Questions in Databases.* s.l., IEEE.

Islam, M. S. & Liu, C., 2016. Know your customer: computing k-most promising products for targeted marketing. *The VLDB Journal.*

Islam, M. S., Zhou, R. & Liu, C., 2013. *On Answering Why-not Questions in Reverse Skyline Queries.* s.l., IEEE.

Islam, M. . S., Zhou, R. & Liu, C., 2013. *On Answering Why-not Questions in Reverse Skyline Queries.* Washington , IEEE International Conference on Data Engineering (ICDE 2013).

Jagadish, H. V. et al., 2007. *Making database systems usable.* s.l., s.n.

Kalyvas, C. & Tzouramanis, T., 2018. A Survey of Skyline Query Processing. *Cornell University Library.*

Liu, Q. et al., 2016. Answering why-not and why questions on reverse top-k queries. *VLDB Journal,* 25(Research Collection School Of Information Systems), pp. 867-892.

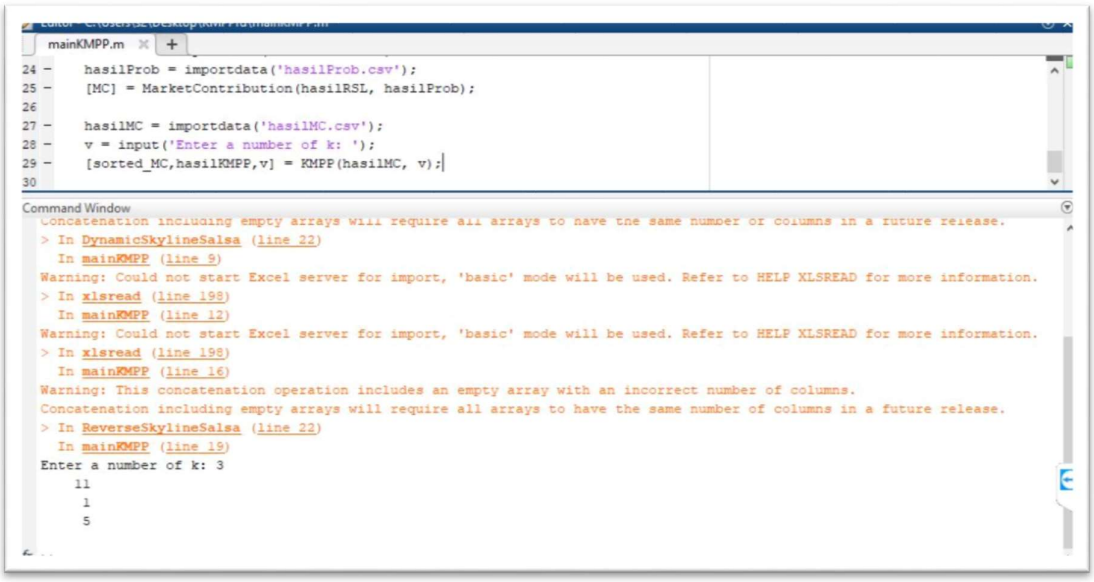
Liu, Q. et al., 2016. Finding Causality and Responsibility for Probabilistic Reverse Skyline Query Non-Answers. *IEEE Transactions on Knowledge and Data Engineering,* 28(11), pp. 2974-2987.

Papadias, D., Tao, Y., Fu, G. & Seeger, B., 2003. An optimal and progressive algorithm for skyline que. *SIGMOD,* pp. 467-478.

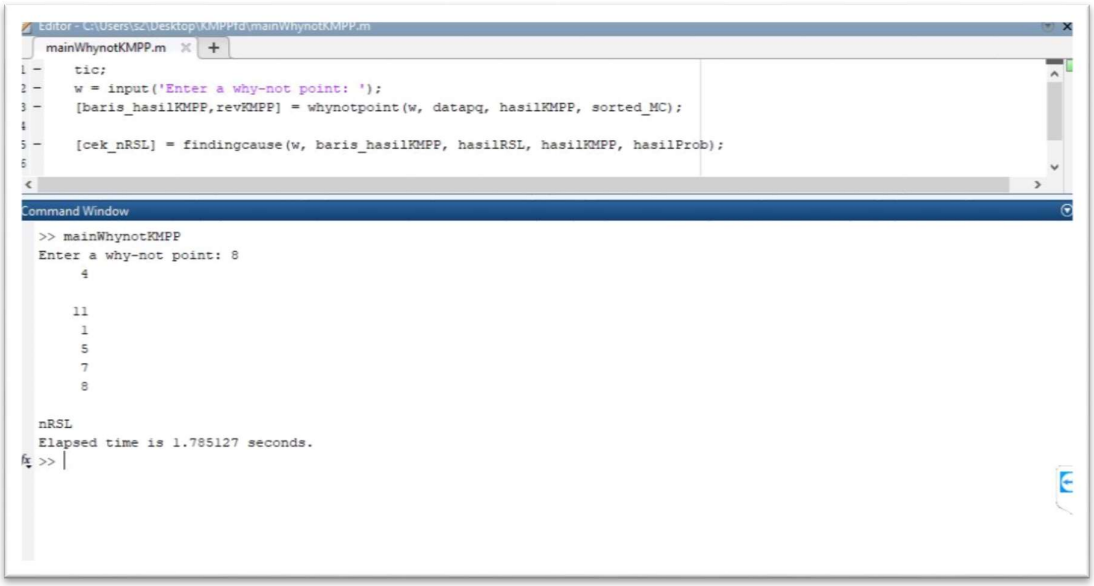
Wu, X. et al., 2009. Finding the influence set through skylines. *EDBT,* pp. 1030-1041.

[Halaman ini sengaja dikosongkan]

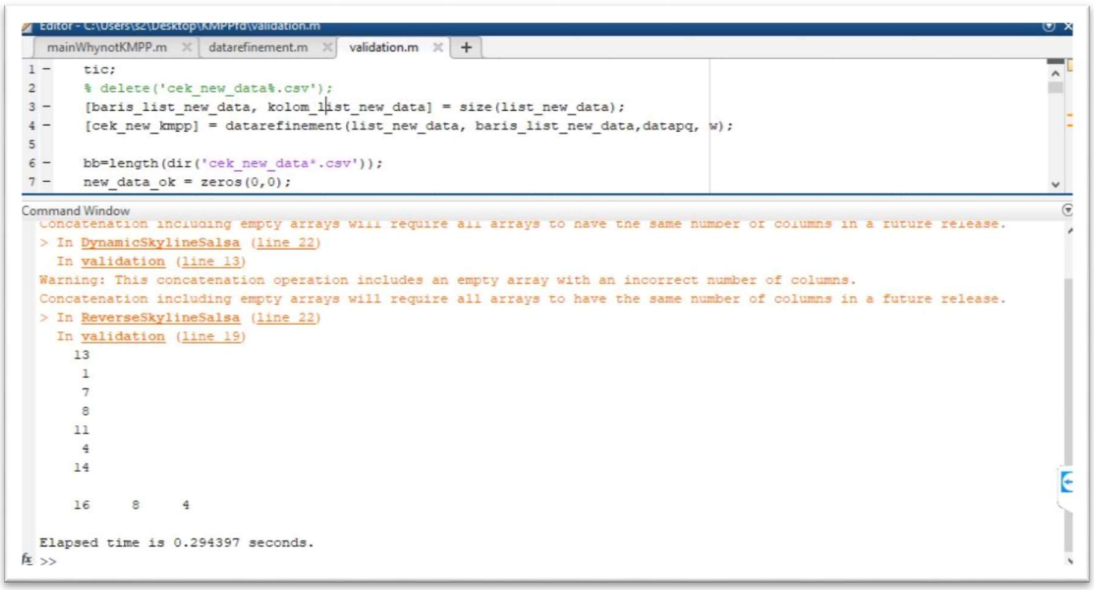
LAMPIRAN



Gambar 7.1 Hasil Proses Komputasi $K - MPP$



Gambar 7.2 Hasil Proses Komputasi *why-not* $K - MPP$



Gambar 7.3 Hasil Proses Validasi