A DEVELOPMENT OF SPATIAL SKYLINE QUERY BASED ON SURROUNDING ENVIRONMENT FOR DATA STREAMING USING APACHE-SPARK

RADEN MUHAMAD FIRZATULLAH

GRADUATE SCHOOL
BOGOR AGRICULTURAL UNIVERSITY
BOGOR
2019
2. Dilonggaran mengunakunin dalam memperdoyangan sebogian dhuwun selunun barya tilik in dholam bentuk opuun lampana ijin IPB.
3. Pengujiowan hokok merugikan reparasiu ung wolder IPB.
4. Pengujiowan hokok unthik reparasiu penuukuupun penuulkan panuukkan hokokra in dilmchi, penuntun hokokra, penuntun hokokra.
5. Dilonggaran mengunakunin dalam sebogian dhuwun selunun barya tilik in lampa meruntun jipun dhuwun ijin.

Hok Cipta Dilindungi, Lindong-Lindong

Bogor Agricultural (Institut Pertanian Bogor)
DECLARATION OF ORIGINALITY AND COPYRIGHT TRANSFER

I hereby declare that the thesis A Development of Spatial Skyline Query Based on Surrounding Environment for Data Streaming Using Apache-Spark is my own work under the supervision of Dr. Eng Ir Taufik Dja'tna, STP, MSi and Dr. Eng Annisa, SKom, MKom. It has never been published in any university. All incorporated originated references from other published as well as unpublished papers are stated clearly in the text as well as in the reference lists.

Hereby, I delegate that the copyright thesis is transferred to Institut Pertanian Bogor University.

Bogor, July 2019

Raden Muhamad Firzatullah
NIM G651170031
SUMMARY

RADEN MUHAMAD FIRZATULLAH. A Development of Spatial Skyline Query Based on Surrounding Environment for Data Streaming Using Apache-Spark. Supervised by TAUFIK DJATNA and ANNISA.

Previous studies of spatial skyline queries based on the surrounding environment posed a challenge in handling skyline searches that supported mobile users. This study introduces a method which allows users to search for spatial objects dynamically. Online streaming data services are currently available to support user movements. With this condition, streaming data requires a longer execution time in processing. This work examines the effectiveness and efficiency of the Apache-Spark framework in developing spatial skyline queries based on the surrounding environment algorithm for data streaming. Further implementation of this algorithm on mobile devices can provide better location access for users.

Comparative analysis of processing time was carried by comparing single processing, parallel distributed computing, and cluster computing algorithms with various measurement evaluation parameters. Result of the test on "number of types in surrounding environment" parameter shows that the execution time of parallel and cluster computing is faster than single computing, 63.694 and 72.772 times faster, respectively. Test on the "raw data size" parameter indicates that parallel computing execution time is 46.553 times faster than single computing, while cluster computing is 74.187 times faster than single computing. In the "main facility search radius" parameter, cluster computing is 45.224 times faster than single computing, and parallel computing is 77.022 times faster than single computing. Lastly, the "number grid" parameter indicates that the execution time for both parallel and cluster computing is faster than single computing, 157.944 and 276.251 times faster, respectively.

Keywords: apache-spark, cluster computing, data streaming, parallel computing, skyline query, spatial data.
RINGKASAN

RADEN MUHAMAD FIRZATULLAH. A Development of Spatial Skyline Query Based on Surrounding Environment for Data Streaming Using Apache-Spark. Dibimbing oleh TAUFIK DJATNA dan ANNISA.

Penelitian sebelumnya tentang spatial skyline query based on surrounding environment meninggalkan tantangan dalam menangani pencarian skyline yang mendukung pergerakan pengguna. Penelitian ini memperkenalkan metode yang memungkinkan pengguna untuk mencari objek skyline spasial secara dinamis. Layanan data streaming online yang tersedia mendukung pergerakan pengguna. Dengan kondisi ini, penggunaan data streaming memerlukan waktu eksekusi yang lama dalam pemrosesannya. Penelitian ini meneliti efektivitas dan efisiensi kerangka kerja Apache-Spark dalam mengembangkan spatial skyline query based on surrounding environment untuk memproses data streaming. Implementasi lebih lanjut dari algoritma ini pada perangkat mobile dapat memberikan akses lokasi yang lebih baik bagi pengguna.

Analisis komparatif waktu pemrosesan algoritma dilakukan dengan membandingkan pemrosesan tunggal, komputasi paralel terdistribusi, dan algoritma komputasi klaster dengan berbagai parameter evaluasi pengukuran. Hasil pengujian pada parameter "number of types in surrounding environment" menunjukkan bahwa waktu eksekusi komputasi paralel dan klaster lebih cepat dibandingkan dengan komputasi tunggal, masing-masing 63,694 dan 72,772 kali lebih cepat. Uji coba pada parameter "raw data size" menunjukkan bahwa waktu eksekusi komputasi paralle 46,553 kali lebih cepat dibandingkan komputasi tunggal, sedangkan komputasi klaster 74,177 kali lebih cepat dibandingkan dengan komputasi tunggal. Dalam parameter "main facility search radius", komputasi klaster 45,224 kali lebih cepat daripada komputasi tunggal, dan komputasi paralle 77,022 kali lebih cepat daripada komputasi tunggal. Terakhir, parameter "number of grid" menunjukkan bahwa waktu eksekusi untuk komputasi paralle dan klaster lebih cepat daripada komputasi tunggal, yakni 157,944 dan 276,251 kali lebih cepat.

Keywords: apache-spark, cluster computing, data streaming, parallel computing, skyline query, spatial data.
A DEVELOPMENT OF SPATIAL SKYLINE QUERY BASED ON SURROUNDING ENVIRONMENT FOR DATA STREAMING USING APACHE-SPARK

RADEN MUHAMAD FIRZATULLAH

Thesis
as partial fulfillment of the requirements
for the degree of Master of Computer Science
in the Computer Science Study Program

GRADUATE SCHOOL
BOGOR AGRICULTURAL UNIVERSITY
BOGOR
2019
Examiner: Dr. Ir Sri Wahjuni, MT

**Bogor Agricultural**

**Hak Cipta** Diliudungi Lindung-Lindung

1. Dilarang mempublikasikan secara selektif Korpa linkis di luar kumpulan dan menyebarluaskan informasi.
2. Dilarang mempublikasikan secara selektif Korpa linkis di luar kumpulan tanpa izin IPB.
3. Penulisan karya merujuk ke publikasi yang wolof IPB.
4. Penulisan karya merujuk ke publikasi yang wolof IPB.
5. Penulisan karya merujuk ke publikasi yang wolof IPB.
Thesis Title: A Development of Spatial Skyline Query Based on Surrounding Environment for Data Streaming Using Apache-Spark

Name: Raden Muhamad Firzatullah
Student ID: G651170031

Approved by
Supervisors

Dr Eng Ir Taufik Djiatna, STP, MSi
Supervisor

Dr Eng Annisa, SKom, MKom
Co-Supervisor

Acknowledged by

Head of Computer Science Study Program

Dr Imas Sukaesih Sitanggang, SSi, MKom

Dean of Graduate School

Prof Dr Ir Anas Miftah Fauzi, MEng

Examination Date: 13th May 2019
Passed Date: 18 JUL 2019
Alhamdulillahi rabbil ‘alamin, Praise is merely to the Almighty Allah SWT for the gracious mercy and tremendous blessing that enables me to accomplish this master thesis entitled: A Development of Spatial Skyline Based on Surrounding Environment for Data Streaming Using Apache-Spark. This master thesis is presented to fulfill one of the requirements in accomplishing S-2 Degree in Computer Science Study Program, Faculty of Mathematics and Natural Sciences, Institut Pertanian Bogor University.

I would like to express my special appreciation to my first advisor, Dr. Eng. Ir. Taufik Djatna, M. Si. and Dr. Eng. Annisa, S. Kom, M. Kom for their valuable assistance and inspiration to the completion of this master thesis. I would also addressed my appreciation to my beloved big family of S2 Computer Science Department 2017. And also thank you to my friends who together worked on the thesis in a guidance commission including Noufal Trezandi, S. Kom, I Made Ary, S. Kom, Gain Motikhon, S. Kom, Yaumil Sitta, M. Kom, Anis Fitri Nur Masruraih, M. Kom and Dewi Asiah Shofiana, M. Kom.

The greatest honor and appreciation would be finally dedicated to my beloved parents, my father Drs. H. Sopiyan, M. Pd and the late my mother Dra. Hj. Ida Zubaedah. It is truly undoubted that loves, cares, spirits, motivation, patience and willingness to wait for my graduation and timeless prayers during days and nights are everything for me. My sincere thanks and love are also dedicated to my beloved auntie and uncle, Evi Rosviawati, Tina Rostina, Tika Rostikawati, S.E, Dani Ramdani, S.T, Tike Rostikewati, S.E, Gigin Ginanjar, and Iing Irwangga, S.H thank you so much from my deepest heart to you all. And also thanks a lot to my cousin Zaenal Arifin, S.Si, M.Si and Indry Prihartuti, S.Si, M.Si who support me when my enthusiasm for working on the thesis was down. Besides that, thank you to my friends who always motivated me to do my thesis, Ridho Dani Ardhana, Emil Dimas Hutomo, S.E, Yudi Syahputra, S.T, Fajar Heirizwan, S.Pt, Andika Kurniawan, Muhammad Ilyas, A.Md, Herry Prayitno, A.Md, Maulana Ikhsan, S.Hut, Bobi Paisal Baraba, S.Pd, Mohammad Hanifah Nur Shafrudin, S.Pd and Ruby Eka Prawira, S.Pd.

Bogor, July 2019

Raden Muhamad Firzatullah
# TABLE OF CONTENTS

**TABLE OF CONTENTS**

v

**LIST OF TABLES**

vii

**LIST OF FIGURES**

vii

**LIST OF APPENDIENCES**

viii

1. **INTRODUCTION**
   - Background 1
   - Problem Statement 2
   - Objectives 2
   - Benefit 2
   - Boundaries 2

2. **RELATED WORK**
   - Skyline Query 3
   - Sort Filter Skyline 4
   - Spatial Skyline Query Based on Surrounding Environment 4
   - Parallel & Cluster Computing 5
   - Data Streaming 7
   - Apache-Spark 9
   - REST API 11
   - Cloud Computing 12
   - Google Maps 13

3. **METHODOLOGY**
   - Research Framework 14
   - Data 14
   - Research Tools 15
     - Spatial Skyline Based on Surrounding Environment & Streaming Process 15
     - Method of Measuring Distance Between Objects 19
     - System Architecture 20
     - Research Phase 21
     - Client Application on Mobile Platform Schema 21
     - Computation Software on Server Architecture 22
     - Integrate Mobile Application and Server Using API 22
     - Topology 23
   - Evaluation Indicator 24
   - Evaluation Scenario 25

4. **RESULT AND DISCUSSION**
   - Single Computing 26
   - Parallel Computing 28
   - Cluster Computing 31
LIST OF TABLES

1. Data Column Description 14
2. Non-Spatial Attribute 15
3. Minimum Hardware Specifications for Mobile Apps Development 15
4. Minimum Hardware Specifications for Computing Cluster 15
5. Evaluation Parameter 24

LIST OF FIGURES

1. Parallel Computing Architecture 5
2. Multi-core processor 5
3. Cluster Computing Architecture 6
4. Cluster computing architecture 6
5. Streaming Data Processing 7
6. Sliding Windows on Data Streaming 8
7. Apache-spark Architecture 9
8. Apache-spark RDD processing 10
9. REST API Architecture 11
10. Cloud Service Model 12
11. Research Framework 14
12. POI Distribution 16
13. (a) Grid and (b) POI on Grid 16
14. Radius of Streaming Area and Finding Main Facility Area 17
15. Streaming Area and Finding Main Facility Area 17
16. (a) POI on Streaming Area and (b) Grid of Streaming Area 18
17. Distance between main facility with surrounding facility 18
18. Buffering Data Condition and Pseudocode 19
19. Haversine Formula on Globe 19
20. System Architecture 20
21. Research Phase Flow Diagram 21
22. Client Application on Mobile Platform Schema 21
23. Computation Software on Server Architecture 22
24. API Architecture 23
25. Topology 23
27. Single Computing: Raw Data Size 26
28. Single Computing: Distance 27
29. Single Computing: Type of Surrounding Facility 27
30. Single Computing: Number of Grid 27
32. Parallel Computing: Facility Attribute Dimension 28
33. Parallel Computing: Distance 29
34. Parallel Computing: Number of Grid 29
35. Parallel Computing: Raw Data Size 29
36. Parallel Computing: Main Facility Search Radius 30
37. Parallel Computing: Type of Surrounding Facility 30
LIST OF APPENDICES

1. Example of Calculating Haversine Formula 43
2. Example of Determine Skyline Object Using SFS. 44
3. Example of Spatial Skyline Based on Surrounding Environment 46

1. Cluster Computing: Facility Attribute Dimension 31
2. Cluster Computing: Distance 31
3. Cluster Computing: Number of Grid 31
4. Cluster Computing: Raw Data Size 32
5. Cluster Computing: Main Facility Search Radius 32
6. Cluster Computing: Type of Surrounding Facility 32
7. Speed up factor: Type of surrounding facility 33
8. Speed up factor: Main facility search radius 34
9. Speed up factor: Number of grid 34
10. Speed up factor: Raw data size 35
11. Speed up factor: Facility attribute dimension 35
12. Speed up factor: Distance 35
13. (a) Configuration Interface and (b) Maps Interface 36
14. Details Interface 37
1 INTRODUCTION

Background

Spatial skyline based on the surrounding environment was explained by Zhiming et al. (2013) as a method to determine skyline spatial objects from a set of spatial object by considering its surrounding spatial objects. Previous method performed on the search for spatial skyline objects only consider when the user is in a fixed position. Based on the shortcomings of the previous algorithm, this study introduces a technique with spatial skylines based on the environment algorithm as the main idea but supports mobile users as an improvement.

Spatial skyline algorithm based on the surrounding environment requires a location data source for spatial objects and non-spatial attributes. The static data is likely for non-mobile users since the spatial object location does not depend on user movement. However, when there is a movement of the user, dynamic data sources are required to provide spatial object data continuously, which changes according to the user's movements. Streaming data described by Namiot (2015) as data that is generated continuously by one or more sources, which are continuous, changes rapidly, and has a large size. Streaming data used in this study is in the form of Points of Interest (POI) locations along with spatial and non-spatial attributes. POI is explained by Yuan et al. (2013) as the location of a particular point that is useful or interesting for someone. User activity tends to be with lower costs and a higher service level. In previous research, streaming data has been used as a data source in the application of skyline algorithms and spatial object searches. Research by Brandt et al. (2017) introduces streaming data processing requests on mobile objects with a Data Stream Management System (DSMS). This proposed method can process queries on mobile objects in DSMS. The test on this study only applies data from ship movements in the sea. However, ship movements are not as complicated as land vehicles having more obstacles. Regarding these researches, this study uses streaming data as the data source for the spatial object, which acts as the user's movement.

Perspective of streaming data is very different from static data. Static data does not have any connection between the dynamic time of initial processing and subsequent processing. Zaharia et al. (2015), stated that data streaming needs fast data processing since the data stream continuously and changes in a time window or radius. Slow streaming data processing affects the relevance of the results obtained (Sarkas et al. 2008). In the research by Zhiming et al. (2013), increase in parameter size results in increased processing time. Apache-Spark explained by Zaharia et al. (2015) as a framework for data streaming processing that applies parallel cluster computing and distributed data handling that is tolerant towards errors. Parallel Computing described by Sechin (2016) as a computer programming technique that allows multiple simultaneous execution/operations CPU (Central Processing Unit). Moreover, Cluster Computing is a group of computers that are interconnected and work together in performing computational processes as a single system (Zaharia et al. 2010). This research uses Apache-Spark to act as a management framework for the operation of distributed computing, including parallel computing and cluster computing in handling real-
time streaming data processing. Previous research by Reyez-Ortiz et al. (2015) shows that Apache-Spark has advantages in computational processes and accuracy compared to OpenMP Beowulf in machine learning processing. Apache-Spark is deployed in this research as a method in processing data streaming. Based on the reasons previously explained, this study introduces a spatial skyline based on surrounding environment algorithm in the case of streaming data using Apache-Spark. Algorithm implementation on mobile devices to provide better location access for users will be tested on a small scale.

**Problem Statements**

The research problem formulation is related to the development of the Skyline Spatial Based Query Environment for Streaming Data using Apache-Spark, namely as follows:

1. How to process spatial skyline on spatial skyline query based on surrounding environment algorithm in the case of data streaming?
2. How to implement Apache-Spark on the spatial skyline query based on surrounding environment algorithm in data streaming processing?
3. How to develop an application that can show the performance of spatial skyline queries based on surrounding environments on the Android platform?

**Objectives**

Based on the formulation of the problem, the objectives in this research focus on the points below:

1. To implement spatial skyline queries based on surrounding environment algorithm in streaming data.
2. To implement Apache-spark in spatial skyline search using the spatial skyline query based on the surrounding environment in streaming data.
3. Apply spatial skyline query based on surrounding environment algorithm using Apache-Spark in the mobile platform.

**Benefit**

This study is expected to contribute benefits by introducing an improved effective method in spatial skyline queries processing based on the surrounding environment using Apache-spark, with streaming data as the data source.

**Boundaries**

Appertaining to the identification of problems, the authors are more directed and in line with the objectives of the study. Some limitations set out in this research include:

1. Algorithm used is spatial skyline query based on the surrounding environment in streaming data.
Google Maps Application Programming Interface (API) used as a streaming data source. Facility preferences and the type of surrounding environment facility used for spatial skyline search is the types of places provided by Google Places. Non-spatial data of each facility on a mobile application are non-spatial data provided by the Google Maps API, while non-spatial data for testing the performance of the algorithm are generated data. The proposed algorithm cannot adapt to the speed of user’s movement.

2 RELATED WORK

Skyline Query

Borzsonyi et al. (2001) first introduced the skyline operator which is operated on large database, where skyline is the extension of SQL queries. In this research, two fundamental skyline query algorithms were introduced, namely Block Nested Loops (BNL) algorithm and Divide & Conquer (D & C) algorithm. Chomicki et al. (2003) introduced the Sort Filtering Skyline (SFS) algorithm which is an improvement of BNL algorithm, where the BNL calculates entropy and ranks skyline candidate based on entropy. The comparison process is not carried out on all candidates that the complexity can be reduced. Furthermore, Bartolini et al. (2006) introduced the Sort and Limit Skyline algorithm (SaLSa) which ranks skyline candidate based on maximum or minimum value and calculates the attribute total value in each object and limits the search for skyline object, so the calculation process is not performed on the whole tuple.

Papadias et al. (2003) introduced Dynamic Skyline Query (DSQ). This research proposed a method for finding skyline object which has spatial and non-spatial attributes at a certain radius where the process of finding the skyline object is not only influenced by spatial attribute but also the distance between the center point and the location of the spatial object. Sharifzadeh and Shahabi (2006) introduced Spatial Skyline Query (SSQ) algorithm which is an improvement of DSQ algorithm, where the skyline points will be considered and the distance in each coordinate is calculated by using Convex Hull and Voronoi Diagram. Therefore, the process of finding a skyline object does not depend on the central point. Zhiming et al. (2013) introduced Spatial Skyline Based on Surrounding Environment algorithm which is an improvement of SSQ algorithm. In this algorithm, the determination of skyline object is not only determined by spatial and non-spatial attributes but also by spatial and non-spatial attributes of surrounding object located in the main object environment. There were several other researchs studied about spatial skyline problem such as in Refs. Kodama et al. (2009), Guo et al. (2010), Lin et al. (2013), Agarwal et al. (2014), Arefin et al. (2014), Annisa et al. (2015), Annisa et al. (2016), Annisa et al. (2017).
Sort Filter Skyline

Sort Filter Skyline explained by Chomicki et al. (2003) is an improvement of Block Nested Loop (BNL) algorithm by presorting the data input ascending based on the entropy function where the value has been normalized. Presorting enforce that a point \( p \) dominating another point \( q \) will be visited before \( q \). This ensures the progressive behavior of SFS and thereduction of the number of pairwise comparisons between points. Algorithm examines the point in the order of the entropy score and saves the buffer into memory and determines the skyline candidate point in the same way as the BNL method. The steps in SFS method are as follow:

1. The first step is normalizing the values in each dimension of the skyline object attribute.
2. Next, calculate the entropy value for each object with the following formula:

\[
E_0(h_i) = \sum ln(h_i + 1)
\]

(1)
3. The next step is sorting based on the smallest entropy value.
4. The last step is removing candidates which are dominated by other object.

In this research, the SFS method is used to determine the skyline object from the remaining skyline candidates after the spatial skyline query based on surrounding environment process is performed. The examples of manual way in determining skyline object using the SFS method are listed in Appendix 2.

Spatial Skyline Query Based on Surrounding Environment

The previous research conducted by Zhiming et al. (2012) is the basis of the study on Spatial Skyline Query Based on Surrounding Environment. This research introduced the method to search the real estate by considering the spatial attribute of object around it but it has not considered the non-spatial attribute that the object has.

Zhiming et al. (2013) proposed a method for choosing spatial skyline points that considered spatial and non-spatial attributes and also considered spatial objects in the surrounding environment. The steps of spatial skyline query based on surrounding environment methods are as follow:

1. Choose the type of facility that will be determined by the object's skyline, and choose the type of surrounding environment facilities that will be considered.
2. Create a grid with \( n \times n \) size specified by the user in defined search area.
3. Furthermore, grouping the main facilities and surrounding facilities on each grid that has been set.
4. Determine the best value of facility attributes in each grid.
5. Determine the search area of the main facility to be searched.
6. Determine the main facility of each grid within the search area.
7. Establish the skyline of the main facility area based on the spatial and non-spatial attribute values.
8. After getting the skyline facility candidate from the previous stage, skyline is searched using the Short Filtering Skyline (SFS) algorithm. In this study, the Spatial Skyline Query Based on Surrounding Environment Algorithm is used as the basis of the main algorithm in determining the best modified POI and it can process POIs when the user moves. The examples of manual way in determining the skyline spatial object using the skyline query based on surrounding environment method are listed in Appendix 3.

**Parallel & Cluster Computing**

Parallel Computing described by Sechin (2016) as a computer programming technique that allows multiple simultaneous execution/operations CPU. In parallel computing system, complex computing processes are divided into process partition and distributed to each processor to be processed simultaneously as illustrated in Figure 1.

![Figure 1 Parallel computing architecture explained by Sechin (2016)](image1)

Parallel computing system utilized multi-core on each CPU to process the dataset by distributing data partition to each processor core, which is then processed in parallel as shown in Figure 2.

![Figure 2 Multi-core processor explained by Sechin (2016)](image2)

Cluster Computing as explained by Zaharia *et al.* (2010) is a group of computers that are interconnected and work together in performing computational process as a single system. Cluster computing has nodes that are interconnected with each other through fast local network. Each node runs its individual operating instances. In cluster computing system, the root computer will divide
the process to the slave computer proportionally in a distributed manner so the process is carried out simultaneously. After the process is completed, the process that has been done on each slave computer will be combined on the root computer, so the process carried out on the cluster system can be accessed on the root computer as in Figure 3.

Figure 3 Cluster computing architecture explained by Zaharia et al. (2010)

Processing dataset in computing cluster is done by splitting dataset into partitions which is then broken down into dataset sections. Then the dataset partition is processed on each worker node connected to the master node on a particular network connection. The worker’s node works on dataset partition by using resource on a single worker node. After that, the dataset partition on each worker node is merged on the master node to be displayed again as a complete result dataset as shown in Figure 4.

Figure 4 Cluster computing architecture explained by Zaharia et al. (2010)

The previous research conducted by Jia et al. (2015) introduced a framework called GeoSpark, where this framework utilizes parallel computing processing and cluster computing in processing large-sized spatial data by utilizing Spatial RDD Layer and Spatial Query Processing in Apache-Spark.
Research conducted by Zhang et al. (2015) performs frequent itemset mining of essential steps of the process of association rule mining. This study proposes an efficient frequent itemset mining algorithm (DFIMA) which applies a matrix-based pruning approach. This research uses Apache-Spark as a framework of cluster computing and parallel computing to increase recurrent computing efficiency.

The research conducted by Thong and Son in 2016 applied computing cluster in fuzzy clustering process of image process. The pattern recognition uses the Fuzzy C-Means algorithm where the camera will capture image in real-time and then the computing device will process pattern recognition from the incoming image. The server device uses a distributed processing environment that is cluster computing on Hadoop MapReduce. Some algorithms in pattern recognition are tested in the introduced method; the result showed the effectiveness of the use of computing cluster. The research conducted by Najafabadi et al. (2015) uses Hadoop MapReduce's distributed parallel computing in conducting big data analytic using deep learning algorithm on large data. In this research, some characteristics of the data were tested on algorithm including data streaming, data with high dimension, and data that had many relationships. Distributed computing and scalability of model showed the performance and good result. The lack of the study was described by data sampling, modeling domain adaptation, semi-supervised learning and not fully monitored by active learning algorithm. In this study, parallel computing and computing cluster are used in handling the increasing of processing time of streaming data, so streaming data process can run effectively, efficiently and get relevant result.

**Data Streaming**

Data streaming is data generated continuously by one or more sources that have continuous, regular characteristic, change rapidly, and have a large size as explained by Li et al. (2014), where the data streaming requires a fast data processing response and can process data that flows in real-time.

![Figure 5 Streaming data processing Aggarwal et al. (2003)](image-url)
window continuously cuts off the part of the data streaming at specific period or condition. It only considers the sliding window element during processing as shown in Figure 5, where the sliding window has the concept of FIFO (First In First Out) as described by Tiziano et al. (2016), the sliding window data that has been received must be processed immediately before the window data in the next period or condition comes as in Figure 6. The delay in processing the data window will result the irrelevant information and not as expected as explained by Aggarwal et al. (2003). Based on this study, it will use streaming data as a data source.

Based on the previous research, streaming data has been used as a data source of the application of distribution computing as conducted by Li Xiao et al. (2014) which use data streaming in a cloud computing environment using parallel programming with Simple Parallel Model (SPM), Alternative Parallel Model (APM), and Distributed Parallel Podel (DPM) to search the skyline of the streaming data. From the paper, the SPM, APM, and DPM models can overcome uncertain sliding window in streaming data running on the skyline query algorithm. As for the lack of the parallel computing model introduced, parallel computing model developed has not been able to handle the increase of the sliding window size efficiently. Thakur et al. (2015) used data streaming as a source of the spatiotemporal data platform in obtaining Geospatial intelligence. The method introduced could process spatial data into real-time geospatial intelligence. In this research, website-based software development is carried out, but mobile-based software that allowed the detection of real-time geospatial data has not been developed.

The research conducted by Bello-Orgaz et al. (2016) introduce the method in mining social network data, where mined data comes from social network streaming data including Youtube, Twitter, Instagram, LinkedIn, and Wordpress. The advantages of this research introduce an effective and efficient method of processing social network data using the Hadoop MapReduce framework. This research used a variety of distributed computing frameworks and databases that are used but did not explicitly explain which framework is the best. Research conducted by Krawczyk et al. (2017) surveyed the use of the Ensemble Learning method in large data streaming handler. The study tried various methods of Ensemble Learning to adapt to the condition and characteristic of the data so that method can be concluded in handling specific data characteristic. The weakness of this research is only discussing the Ensemble Learning method in the case of classification and regression, but other instances such as clustering or clustering mixed with classification is not explained. Yun and Lee (2015) introduced a method in data streaming that allows deleting pattern so the condition on data.
stream can be shifted to the next streaming window without affecting the result obtained. The performance of the proposed algorithm with a sophisticated tree-based approach concerned various real and synthetic data sets. The experimental result showed that the method in this study was more efficient and measurable than competitor in terms of runtime, memory, and pattern making. In this study, streaming data is used as a dynamic data source consisting of spatial and non-spatial data from spatial objects in overcoming user movements.

Apache-Spark

Apache-spark as explained by Zaharia et al. (2016) is an open-source cluster computing framework where data is distributed to each node in the cluster to be processed in parallel, where the process of distributing data is done with the Resilient Distributed Dataset (RDD) architecture as shown in Figure 7. RDDs are multiset read-only data that are distributed to node member on a cluster of cluster nodes, where data processing is evaluated by the master node so it is tolerant of error. Spark uses the RDD architecture in data distribution, where data will be processed in a single work unit and random access memory and computer memory cache. Spark conducts repeated evaluations, so the data distribution process to each node cluster is tolerant of error.

Figure 7 Apache-spark architecture (Apache-spark 2018)

The processing dataset in Apache-Spark as explained by Zaharia et al. (2016) is done by converting the full dataset into RDD (Resilient Distributed Datasets), where RDDs will be broken down into RDD partition which is then processed on each slave node. The Apache-Spark program script has been embedded in each slave node, so RDD processing can run simultaneously. After the RDD is processed, the RDD partition is recombined again by the master node and returned to the whole RDD. In Apache-Spark, the steps in converting datasets to RDD and manipulating RDD are called Transformation Process, while the RDD merging process of each worker node is called Action Process. The Action Process has a higher complexity and processing cost than the Transformation Process. RDD result from the merger can be recalled in form of JSON or dataset as shown in Figure 8.
In the previous research, Apache-Spark has been used in large-scale data distributed computing to handle the increase in computational time algorithm as done by Wang et al. (2017) they used parallel computing based on MapReduce in handling the processing of large spatial skyline dataset but in its application not all spatial objects are taken into account, spatial objects that do not have complete attributes will be eliminated, resulting in changes in accuracy obtained. The research done by Chen et al. (2016) used Apache-Spark on cloud computing in processing Parallel Random Forest (PRF) algorithm for large dataset shows the advantage of computational time in classification, performance, and scalability but the proposed research method has not been able to handle computing on data streaming.

Research by Gopalani and Arora (2015) compared the Apache-Spark and MapReduce framework in processing big data with the K-Means algorithm. The result of the study showed that Apache-Spark was superior to Hadoop MapReduce in treating the K-Means algorithm. In this study, the drawback is only the usages of big static data, which does not use dynamic data or streaming data so the performance of MapReduce and Spark in processing big data and data streaming has not been prominent. Koliopoulos et al. 2015 introduced a framework that combines Apache-Spark with Weka. The study compares Weka’s performance with single Weka process that run on the Apache-Spark framework in processing large data. The result showed that Weka which is run on the Apache-Spark framework has 4x faster speed than Weka that run in a single environment. In this study, the drawback was only testing the performance of the proposed method in large-scale static data but not testing it in a dynamic data environment. In this study, Apache-Spark is used as a distributed streaming data processing framework in cluster computing and parallel computing environment.

Figure 8 Apache-spark RDD processing (Apache-spark 2018)
REST API

Representational State Transfer Application Programming Interface (REST API) as described by Yates et al. (2014) is a website-based communication architecture standard which is often applied in developing website-based computing service. Generally, the REST API uses HTTP (Hypertext Transfer Protocol) as a protocol for data transfer communication between client and server. In REST architecture, the REST server is in charge of providing resource (data source) and while the job of REST is to access and display these resources for further use, such as Figure 9. In the REST API, each resource is identified by URIs (Universal Resource Identifiers) or global ID where the resource is represented in form of text format, JSON or XML. The REST API process is not too affected by bandwidth, which makes REST more suitable for use in cloud computing.

In the previous research, REST API has been used in the application of mobile healthcare information management system that is done by Doukas et al. (2010), the system built mobile device which will provide input in form of medical image which then image recognition processing for medical diagnostic is done using cloud computing. Communication between mobile device and cloud computing uses the REST API protocol. Research conducted by Sukhoroslov et al. (2015) used REST API as an intermediary for scientific publication system and website-based journal with distributed computing. This research separated web application from computing resource and provided remote computing access. The research conducted by Pop in 2016 designed a cloud computing system that run machine learning algorithm on server. Cloud computing have run in distributed computing environment including Spark, CUDA and Dryad. Cloud computing and client are connected by the REST API, where the Macine Learning algorithm is built using the R programming language and Python, and the client can access the result in JSON form. The result of the study showed that cloud computing running in multi-processing environments has better performance. Research conducted by Fabbrizio et al. (2009) introduced architecture in speech recognition processing on cellular device and connected with server processing, in its application using the REST API as a liaison between mobile device and server. The introduced architecture is called AT & T mashup speech; architecture that has good latency in performing speech recognition and transformed the sound in form of text. In this study, the REST API is used as a connecting protocol for mobile application with cloud computing that implements parallel computing and computing cluster.
Cloud Computing

Cloud Computing as explained by Mell and Grance (2011) is a combination of user of computer technology (computing) and internet-based development (cloud) where the computing process will be carried out on an internet basis as a service, so the user simply input data into cloud computing and it will process the data and return the result through the internet.

In the application of cloud computing, the system acts as a backend, which is in charge of processing input and computational data and produces output from the result of processing input. In its application, cloud computing requires an internet network to connect client application (frontend) with computing software in cloud computing (backend), where in general both software was built using different programming languages and is associated with API. The cloud computing model is illustrated in Figure 10.

The research conducted by Chen et al. (2015) applies the initial computational capability that is connected to mobile phone user access, where cloud computing will run multi-user computing in a multi-channel wireless interface environment. In this research, computation with NP-hard complexity is done and got an optimal and effective solution. Research conducted by Zhu in 2017 discussed the use of Hadoop Cloud Computing in data mining process in the case of electric power system. This study used a neural network algorithm in predicting interference with electrical power system. The result of the study showed that the maximum algorithm error is only 0.2% and the data mining performance is feasible and practical to implement. The lack of this study was not discussing specifically the neural network algorithm used, only considering the use of Hadoop Cloud Computing in the implementation of the system built.

Research done by Tawalbeh et al. (2016) applied a mobile cloud computer that integrated cloud computing and mobile device to overcome the limitation of memory and CPU resources in health network application. Cloudlet-based mobile cloud computing infrastructure that will be used for big health data applications. Conclusion is drawn regarding the design of network health care system using large data and mobile cloud computing technology to run effectively and efficiently. The weakness of this study was only explaining the use of mobile cloud computing without discussing algorithm in processing the big data analytics. Research by Cao et al. (2018) used cloud computing and the MapReduce
algorithm in processing data for signal error diagnosis on high-speed train. Classification algorithm is used the diagnosis in streaming data with the high data flow. Streaming data processing is done in cloud computing that has cluster nodes and distributed computing. The result showed that the proposed algorithm can reduce the number of calculation in execution process, significantly reduce memory space consumption, and increase the speed of calculation in the railroad signal system. In this study, cloud computing is used in the operation of server computing, where server computing utilizes cluster computing and parallel computing so the input from the client application will be sent to the application server which is then processed on the server side. After the processing is completed, the cloud would provide results from computing to the client application.

Google Maps

Google Maps (Google 2018) is a web mapping service that provides satellite imagery, street map, traffic condition, and travel route. Google Maps implements the WGS 84 / Pseudo-Mercator coordinate system Battersby et al. (2014) where the returned point is latitude and longitude. Google Maps has a world coordinate database along with spatial objects that are grouped based on the spatial object group. Google Places (Google 2018) is a feature of Google Maps that provides spatial streaming service related to spatial object in specific location, Google Places classifies spatial objects based on place category and provides a spatial object search feature in a stream at a certain radius from the user's location (Google 2018).

The previous research conducted by Branson et al (2017) used the Google Maps streaming data source to detect and recognize tree species in urban area using the convolutional neural network method, the method introduced was able to recognize tree species from various angles through images captured by Google Maps. Research conducted by Mingqi et al. (2016) uses google places as a spatial data source in data mining process to determine the semantic location. In this study, the determination of the physical location of the semantic location will be done by extracting user visit point based on GPS and Google Places. The research conducted by Majid et al. (2012) made tourism destination recommendation based on geotagged social media, where the social media used was Flicker and Google Maps. Tourist activities in several cities of China are processed; the data taken includes photo, comment, and the value of visitor satisfaction. This study resulted in a standardized recommendation method that was able to predict tourist preference in new or unknown cities with more precise accuracy and resulted in better recommendation. The research conducted by Jiang et al. (2014) utilizes social network and Google Maps data in classifying and disaggregating urban area. In this study, Google Maps POI, which was located in the Boston metropolitan area, then analyzed the land use, transport and environment models using the classification method. The disadvantage of this research is that the POI data taken is not streaming data, but rather static data taken beforehand. In this study, we used Google Maps and Google Places as digital maps that display and detect the presence of user as well as dynamic source of spatial data object.
3 METHODOLOGY

Research Framework

In order to accomplish the solution for objectives, this study is based on the research framework as shown in Figure 11.

Data

Google Maps and the Google Places API acts as the data source in this research. Attributes used from the dataset are listed in Table 1.

Table 1 Data column description

<table>
<thead>
<tr>
<th>Column</th>
<th>Explanation</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Longitude</td>
<td>Location of the presence of spatial objects on the longitude line.</td>
<td>Float</td>
</tr>
<tr>
<td>Latitude</td>
<td>Location of the presence of spatial objects on the latitude line.</td>
<td>Float</td>
</tr>
<tr>
<td>Place ID</td>
<td>ID of spatial object.</td>
<td>String</td>
</tr>
<tr>
<td>Place name</td>
<td>Name of spatial object.</td>
<td>String</td>
</tr>
<tr>
<td>Place type</td>
<td>Place type of spatial object.</td>
<td>String</td>
</tr>
<tr>
<td>Non-Spatial Attribute</td>
<td>Non-spatial attribute values of spatial objects.</td>
<td>Float</td>
</tr>
</tbody>
</table>

Non-spatial attributes in each type of places on Google Maps are shown in Table 2. These non-spatial attributes will later be used as preferences in determining the skyline object.
The minimum specifications of hardware used in developing mobile-based applications are listed in Table 3. Minimum hardware specifications used as nodes in the Apache-spark computing cluster is shown in Table 4. The software used in application development includes Android Studio version 3.2.1 minimum version SDK 14 for API as IDE android application development, Jupyter Notebook version 1.0.0 as IDE python in development of the API, Apache-spark version 2.2.0 as framework parallel computing, Anaconda version 4.511 as package and environment management system, Python version 3.6.5 as programming language on software server, JSON as data format which is used on the API, Hadoop version 2.12.7 as Apache-spark programming language, and JDK version 8 as Java code compiler device to bytecode.

### Table 2 Non-Spatial Attribute

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Place Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Price</td>
<td>Hotel, Restaurant, Café, Bar, Bakery, Florist, Store.</td>
<td>The price provided by Google Maps is the average price of all packages/products.</td>
</tr>
<tr>
<td>Rating</td>
<td>Almost all types of places have a rating.</td>
<td>Ratings are obtained from the rating of Google Maps users.</td>
</tr>
<tr>
<td>Opening time</td>
<td>Hotel, Restaurant, Café, Bar, Bakery, Florist, Store, Gas Station, Museum, Dentist, Florist, Pet Shop.</td>
<td>Opening time is entered by the owner of the place to Google Maps.</td>
</tr>
<tr>
<td>Capacity</td>
<td>Stadium, Parking Area, Studio, Cinema.</td>
<td>Capacity is entered by the owner of the place to Google Maps.</td>
</tr>
</tbody>
</table>

### Research Tools

The minimum specifications of hardware used in developing mobile-based applications are listed in Table 3. Minimum hardware specifications used as nodes in the Apache-spark computing cluster is shown in Table 4. The software used in application development includes Android Studio version 3.2.1 minimum version SDK 14 for API as IDE android application development, Jupyter Notebook version 1.0.0 as IDE python in development of the API, Apache-spark version 2.2.0 as framework parallel computing, Anaconda version 4.511 as package and environment management system, Python version 3.6.5 as programming language on software server, JSON as data format which is used on the API, Hadoop version 2.12.7 as Apache-spark programming language, and JDK version 8 as Java code compiler device to bytecode.

### Table 3 Minimum hardware specifications for mobile application development

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM (Random Access Memory)</td>
<td>3GB for software development on Android Studio, 1GB to run the Android emulator.</td>
</tr>
<tr>
<td>HDD Space</td>
<td>2GB.</td>
</tr>
<tr>
<td>Screen Resolution</td>
<td>1280 x 800.</td>
</tr>
</tbody>
</table>

### Table 4 Minimum hardware specifications for the computing cluster

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>Minimum 4 core</td>
</tr>
<tr>
<td>RAM</td>
<td>4GB.</td>
</tr>
<tr>
<td>Network</td>
<td>LAN (Local Area Network)</td>
</tr>
</tbody>
</table>

### Spatial Skyline Based on Surrounding Environment and Streaming Process

Sliding window as explained by Aggarwal *et al.* (2003) is the flow of data on streaming data in a certain period or condition. As an example, a user wants to find cheap hotels close to the supermarkets and restaurants with a good service,
and also want to find supermarkets and restaurants that has competitive prices. Figure 12 showed the distribution of the existence of POI for that particular example.

Figure 12 POI distribution

Hotel is the main facility illustrated by square symbols, $H = \{ h_1, h_2, h_3, \ldots, h_{10} \}$. Restaurant is surrounding facility illustrated by the circle symbol $R = \{ r_1, r_2, r_3, \ldots, r_7 \}$. Supermarket is another surrounding facility illustrated by the triangle symbol $S = \{ s_1, s_2, s_3, \ldots, s_{10} \}$, and user location illustrated by hexagon symbol.

The next step is to build a grid with a size of $n \times n$ and provide a label id on each grid as shown in Figure 13(a) and 13(b). Then, set the radius of streaming and the radius of the search main facility area with the center point of the user's location.

Figure 13 (a) Grid and (b) POI on grid

Streaming area is an area that includes main facility and surrounding facility in a sliding window. Finding area is an area which includes the main facility that are candidates for skyline object in a sliding window. In Figure 14 the streaming radius is a circle with radius $x$ and the radius of the search main facility area is a circle with radius $y$. Only consider the main facilities within the finding area.
radius to be a skyline object. The surrounding facilities that are within the radius of finding area and streaming area that can be included in the sliding window are shown in Figure 15.

Hotels in the streaming area but outside the finding areas such as h₅, h₈ and h₉, are not included in the skyline candidate. The surrounding facilities within streaming area but outside finding areas such as the supermarkets s₄, s₁₀, restaurants r₁, r₂, r₄, r₅ are considered as surrounding environment facilities effecting the selection of the best hotels.

In Figure 16(a) the main facilities considered to be POI/skyline points are facilities that are within the radius of the search area such as h₂, h₃, h₄, and h₇. Meanwhile surrounding facilities that are considered in determining the skyline object are facilities within the radius of the search and streaming area including supermarkets s₄, s₅, s₆, s₈, s₁₀ and restaurant r₁, r₂, r₄, r₅.
Supporting facilities located within streaming area are considered, since there is a possibility these facilities are better than the facilities located in the search area. However, facilities that are outside of streaming area in the grid \{(0,0), (0,1), (0,3), ... (6,6)\} are not included in the sliding window because the facility has a greater distance than facilities in the search and streaming area, shown in Figure 16(a). A set of new buffer sliding window processed for observation on mobile users to search area in different and dynamic facilities in the old search area as part of the new search area and are not in the previous POI/skyline objects.

Determining the distance of facilities around the main facilities is based on a grid that has the main facilities in it as well as the actual grid located around the main grid as in Figure 17. Determination of the minimum distance between the main object and the surrounding object is not smaller than the distance between the main object and the grid around it. Thus, the distance used has a radius of 1.5 from the length of the lattice side. As an example, if a grid has a size of 4 meters,
then the distance from the main facility to the surrounding facilities can be determined to be $1.5 \times 4$ meters, which is 6 meters.

Figure 18 Buffering data condition and pseudocode

Figure 18 shows that the initial step of the user's existence will be checked every time window $(t_i)$ specified. If the user exits from the finding area to either west, east, north or south, the facility buffer in the new search area changes according to the user's current location. Meanwhile, the user's movement will be identified at a certain period as a data buffering trigger if the user moves to an area outside the radius of the main facility's search area. The process of checking the user's location was carried out at a particular time. Figure 18 shows how the triggers update the condition of the sliding window. After the skyline candidate object found in the form of a data line with a skyline determinant attribute dimension, the determination process of the skyline object will use the SFS fundamental algorithm.

Method for Measuring Distance between Objects

Figure 19 Haversine formula on globe
Haversine formula is an equation in navigation, giving the length of the distance between two points on the ball from longitude and latitude. The Haversine formula as described by Brummelein and Robert (2013) is a remarkable case of a more general formula in spherical trigonometry. Figure 19 shows how the law of haversine relate to the sides and angles of a spherical triangle which measures the location between points based on the slope. This study uses haversine formula as the spatial objects taken are sourced from the Google Maps / Places API, where Google Maps is a spatial data streaming service that implements the WGS 84 / Pseudo-Mercator coordinate system (Battersby et al. 2014). Therefore, points will return in the form of latitude and longitude so that the measurement of the distance between points will be influenced by the slope of the earth’s surface, such as the common distance calculation on the globe.

\[
d = 2r \cdot \arcsin \left( \sqrt{\sin^2 \left( \frac{\Delta \varphi}{2} \right) + \cos \varphi_1 \cdot \cos \varphi_2 \cdot \sin^2 \left( \frac{\Delta \lambda}{2} \right)} \right)
\]  

(2)

Haversine formula is determined by latitude (\(\varphi\)) and longitude (\(\lambda\)) from the starting point to the end point to be searched and is determined by the radius of the earth (\(R\)) which has a fixed value (radius = 6.371km). Details and sample concepts are available in Appendix 1.

### System Architecture

In order to implement Apache-Spark, system architecture was developed by combining two different platforms. The server platform which runs the algorithm acts as a computing machine whereas the client platform on an Android device as the user interface. The system architecture is shown in Figure 20. Communication between mobile devices and server devices uses the REST API as a link between two environments. Data communication format uses JSON (JavaScript Notation), both for communication between internal and external environments such as Google Maps.

![System Architecture](image-url)
The research phase flow chart is shown in Figure 21. Explanation of the research phases as follows.

**Client Application on Mobile Platform Schema**

In this research, the first step is building a client application on an Android device, where the client device will be integrated with the Google Maps API in searching for the closest spatial point and displaying the skyline object on a digital map. The client application architecture on an Android device is shown in Figure 22.

The processes in Figure 22 are explained as follows:

1. The client device makes a request on the Google Maps API with some parameters including current location, place type spatial object, and place type spatial object surrounding.
2. The Google Maps API responds to the client device request in JSON form, where the JSON data contains all data spatial objects around the current location.
3. Furthermore, the client device will pass a request by sending the current location parameter (longitude and latitude) to the server API.
Another process that runs on client devices is the process of storing spatial object data from the Google Maps API. The client device will also display the skyline point in form of a digital map using the help of the Google Maps API.

Computer Software on Server Architecture

The second step in this research is building computing software on the server. At this step, the server software computes skyline spatial query based on surrounding environment with parallel computing by using Apache-spark and generate skyline point and coordinates the facilities to be sent in response to client devices. The computing software architecture on the server is shown in Figure 23. The processes described in Figure 23 can be described as follows:

1. API sends a request in form of a longitude and latitude parameter where the user is located to the server software, where communication between the API and the server software uses the HTTP protocol network. The request method used on API is the POST method.
2. The next process, the spatial skyline point coordination from the calculation results is stored to the database.
3. The last process, the spatial skyline point coordinates will be sent to the client device as a response to the API request sent by the client device in the form of JSON.

Integrate Mobile Application and Server Using API

The third step in this research is integrating computational software on server and mobile application with API. Server and mobile devices are developed in different environment. A mobile app is developed by using Java language and XML while the server software is developed using python programming language. Then we need the integration of server software and Android. The API will act as the second integrator of the software described in Figure 24.
The explanation of the processes of API architecture in Figure 24:

1. The client device requests the API using the HTTP post and gets protocol using the REST API, the parameters sent by the client to the API are current location which consist of the longitude and latitude of the user's presence. Data that was initially been float type in Java programming language was encrypted to JSON first.

2. The request is received by the server and the data which initially in the form of JSON will be decrypted into float using Python programming language. Then the data will be processed by the software to produce spatial skyline point coordination that will be sent as a response to the client.

3. Result data will be encrypted again into JSON which previously had a float data type in Python programming language. Furthermore, JSON data is sent as a response to request made by mobile devices. The response received by the client application in form of JSON will be decrypted back into a float data type in Java programming language, then the data will be displayed in form of a spatial point on the Google Maps API.

![API architecture](image)

Figure 24  API architecture

![Topology](image)

Figure 25  Topology

Figure 25 showed the topology that used in this research and it is a hybrid topology, which combines the bus topology with the star topology, where there is one switch and 15 computer worker nodes that are connected to each other. This topology has been applied to the laboratory where the algorithm is tested, so it can no longer be changed. The 15 worker nodes of the computer are connected to each
other on a wired network with a UTP cable as well as a master node that is also connected directly to the switch using a UTP cable. Checking the network is done by sending ping command from the master node to the worker node, which result in low response time of around 5-20 ms.

**Evaluation Indicator**

The final step in this research is evaluation. In this step, a performance evaluation is taken place as the algorithm indicated. Evaluation indicators used in testing the performance of parallel computing and cluster computing as explained by Wilkinson *et al.* (2004) includes Speedup Factor $S(p)$, a measure to compare the performance both on single and parallel processing in the same algorithm, in short it notices both execution running time on a single processor ($t_s$) and execution time on a multiprocessor ($t_p$).

$$ S(p) = \frac{t_s}{t_p} \quad (3) $$

Message-passing computing ($t_p$) is parallel computing execution time, which is caused by communication time factor ($t_{comm}$) and computational time ($t_{comp}$).

$$ t_p = t_{comm} + t_{comp} \quad (4) $$

The next step, testing used parameter variation that affect ed the spatial skyline query algorithm based on the surrounding environment. Parameter configuration and value parameters used in this research can be seen in Table 1.

Based on research conducted by Zhiming *et al.* (2013) the configuration parameters in Table 1 are used in testing the processing time because they have a role in skyline processing time. While the default value parameter chosen is a value that has a not too low but not too high value, so it does not affect the result of processing time in the other parameter value that being tested.

**Table 4 Evaluation parameter**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw data size for each type of facility</td>
<td>500, 1000, 2000, 4000, 8000</td>
<td>500</td>
</tr>
<tr>
<td>Number of type of surrounding environment</td>
<td>2, 3, 4, 8, 16</td>
<td>3</td>
</tr>
<tr>
<td>Number of grid</td>
<td>25, 100, 400, 625, 2500</td>
<td>100</td>
</tr>
<tr>
<td>Number of dimension attribute each type of facility</td>
<td>1, 2, 4, 8</td>
<td>2</td>
</tr>
<tr>
<td>Distance of surrounding facilities from the main facilities in meters</td>
<td>10, 50, 100, 200, 500</td>
<td>100</td>
</tr>
<tr>
<td>The main facility search radius in meters</td>
<td>5, 10, 20, 40, 80</td>
<td>20</td>
</tr>
</tbody>
</table>
Evaluation Scenario

For evaluation of the proposed algorithm, it is necessary to test the performance in 3 different environments such as simulation of conventional skyline algorithm in single computing, simulation proposed algorithm in parallel computing, and cluster computing using Apache-Spark framework. These experiments used synthetic data that contain spatial and non-spatial data that has an anti-correlation data distribution, where data is splitted into a sliding window. Each test processes 50 sliding window iterations and the average processing time is used as execution time. Proposed algorithm were tested using various parameters including the number of raw data, number of types of surrounding facility, number of dimension of non-spatial attribute, the number of grid and finding radius area.

The first evaluation simulated conventional algorithm in a single computing environment. In this evaluation, the algorithm run independently without distributed data on the core processor. In the second evaluation, the proposed algorithm run on Apache-Spark parallel distributed computing on one computer. Apache-spark framework distributes data on the processor and the core processor processing data simultaneously. Parallel computing simulations are carried out on various number of core processor (2,3,4 cores). Single computing and parallel computing simulations are performed on PCs with Intel i7, 2.6 GHz, 4 cored processor, and 4 GB RAM. In the last evaluation, the proposed algorithm runs on Apache-Spark cluster distributed computing on computer cluster environment. The master node distributes data to the worker node, and the worker node processes the data simultaneously, then the data from the worker node will be reunited on the master node. The use of worker nodes varied including 2 nodes (8 cores), 8 nodes (32 cores) and 15 nodes (60 cores) which each worker node has the same specification and operating system. Cluster computing simulations are perform on PCs with Intel i7 processors, 2.6 GHz, 4 cores, 4 GB RAM and 15 worker nodes that have specs such as Intel i5 processors, 3.00 GHz, 4 cores, 4GB RAM and are connected to a LAN (Local Area Network) network. The final step is to compare the performance of each environment and examine the effectiveness and efficiency of the proposed algorithm.

4 RESULT AND DISCUSSION

The initial stage of evaluation in this research evaluated the increasing of the parameters in the skyline query based on surrounding environment spatial algorithm in stable condition and it measures the effect of increasing parameter values on increasing processing time and testing whether data streaming can be applied. The second stage evaluated the use of the algorithm proposed in Apache-Spark which runs in parallel computing. This study observed whether the use of parallel computing in Apache-Spark would increase processing time effectively or not. The last stage evaluated the use of the algorithm proposed in Apache-Spark which ran on computing clusters, in this study observing whether the increase in the number of worker nodes (the number of resources) will increase the algorithm's performance time and prove whether streaming data processing can
run in real-time. From the three results of the evaluation, the value of speed up factor will be calculated whether the performance of cluster computing and parallel computing runs effectively and efficiently

Single Computing

The first evaluation simulates conventional skyline algorithms in a single processing environment. The first experiment measured the time of skyline object search by using variation in number of attribute dimensions, where other parameters have values according to the default values of Table 1. Figure 26 showed the execution time on number of dimension attributes of 1, 2, 4, and 8 dimensions. From these results, it can be seen that an increase in the number of dimensions can affect the increase in algorithm processing time. This is due to the more complex consideration in determining the skyline object.

![Facility Attribute Dimensions](image)

**Figure 26** Single computing: facility attribute dimension

The next experiment tested the algorithm for each raw data size as shown in Figure 27. This was due to the increasing size of raw data, the number of objects considered as candidates for the skyline object increases.

![Raw Data Size](image)

**Figure 27** Single computing: raw data size

The third experiment showed the execution time increase along with increasing distance as shown in Figure 28. This was due to the increasing distance between the main facility and the surrounding facility so the number of surrounding facilities considered which have more result in more complex consideration in determining the skyline object.
The next experiment shows the effect of the number of facilities around it as shown in Figure 29. This happens because the more types of facilities are around, the more attributes that need to be considered in determining the skyline object.

![Figure 28 Single computing: distance](image)

![Figure 29 Single computing: type of surrounding facility](image)

The next experiment showed the effect of increasing the number of grids on processing time. Figure 30 showed that computing time increased with increasing number of grids. This was because an increase of the grids number would also increase the number of clusters made.

![Figure 30 Single computing: number of grid](image)

Figure 31 showed an increase in computational time often with an increase in the main facility search radius. This was due to the greater radius, the main facility that be considered as a skyline object be even more great.
From evaluation carried out in single computing, the increase in parameter values of all parameters can be increased computing time. This indicated that there was an increase in execution time, especially in the large dimension and large number of raw data. It should be overcome so the data processing time ran efficiently.

**Parallel Computing**

The second evaluation was carried out in the parallel computing environment by simulating the proposed skyline based on surrounding algorithm. Tests are carried out on three different conditions, where each condition uses a number of varying core processors including 2, 3 and 4 cores.

The first experiment is done by calculating the skyline object search computation time on each attribute dimension, where an increase in the number of attribute dimensions can be reduced by adding parallel processing usage in parallel computing as shown in Figure 32. The next experiment shows an increase in execution time due to an increase in distance between the main facility and surrounding facility which is inversely proportional to the increase in the number of core processing in the parallel computing environment, as shown in Figure 28.
The third experiment showed an increase in execution time due to an increase in number of grids inversely proportional to the increase in number of core processing of parallel computing environment as shown in Figure 34. Increasing the number of core processing could be reduced the proposed algorithm execution time to increase the number of grid.

The next experiment showed the effect of raw data size on the execution time of the proposed algorithm and can be increased the processing time, but this was inversely proportional to the addition of the number of processing cores, as shown in Figure 35.
The sixth test showed the effect of increasing the main facility search radius on processing time that could be overcome by increasing the number of core processing in proposed algorithm, as shown in Figure 36.

![Figure 36 Parallel computing: main facility search radius](image)

The last experiment showed the effect of increasing the number of types of facilities around it. Processing time increase inversely with the increase in the number of facilities types around it, but an increase in the number of cores reduced the increase in processing time, as shown in Figure 37.

![Figure 37 Parallel computing: type of surrounding facility](image)

From the research conducted on parallel computing, the result of increasing processing time along with increasing parameter could be solved by increasing the number of processor in parallel computing. This was due to an increase in the number of processor, the data will be divided into more processors and processed in a more effective and efficient manner.

**Cluster Computing**

The third evaluation was carried out on the cluster computing environment by simulating the skyline based on surrounding proposed algorithm. Tests are carried out in 3 different conditions, where each condition used varying worker nodes including 2 (8 cores and 6 GB RAM), 8 (32 cores and 24 GB RAM) and 15 (60 cores and 45 GB RAM) worker nodes.
Figure 38 Cluster computing: facility attribute dimension

Figure 38 showed that the execution time increase the dimension of the attribute, but increasing the number of worker nodes in cluster system. It also overcame the increase in execution time because the resource consisting of RAM and core processor was also increase, as shown in Figure 33. The second experiment showed the effect of increasing the distance parameter between the main facility and the surrounding facility. Figure 39 showed an increase in execution time, as the distance increase inversely proportional to the increase in worker nodes in tsystem cluster. This was due to an increase in system cluster resources in form of RAM and core processor obtained from worker nodes.

Figure 39 Cluster computing: distance

Figure 40 Cluster computing: number of grid
The third experiment showed an increase in execution time due to an increase in number of grids inversely with an increase in number of worker nodes in system cluster due to increased resources, as shown in Figure 40.

The next experiment showed that the effect of raw data size on the execution time of the proposed algorithm could increase processing time, but this was inversely proportional to the addition of the number of processing cores, as shown in Figure 41. The five tests showed that the increase in processing time due to an increase in the main facility search radius can be inversely proportional to the increase in worker nodes computing cluster, as shown in Figure 42.
The last experiment showed that the increase in processing time due to an increase in the type of surrounding facility is inversely proportional to the increase in worker nodes in the computing cluster system, as shown in Figure 43.

From the research conducted on computing cluster, the results of increasing processing time along with increasing parameter can be overcome by increasing the number of worker nodes in computing cluster. This was due to an increase in the number of worker nodes which mean an increase in the number of resource processing including CPU and processor. Therefore, processing time in real-time can run effectively and efficiently.

**Speed Up Factor**

The speed up factor in each parameter between single computing with parallel computing showed that the use of parallel computing and cluster computing in proposed algorithm could improve algorithm processing performance. The number of core processor in parallel computing and worker nodes in computing cluster was directly proportional to the increase in processing time.

![Figure 44 Speed up factor: Type of surrounding facility](image)

The increase in speed up factor or acceleration in both environments namely parallel computing and cluster computing compared to single computing is shown in Figure 44. Even though the number type of surrounding environment increased and the acceleration also increased, while cluster computing increased speed up factor and became static when the number type of surrounding environment was more than 8, but the parallel computing was still increasing.
In the main facility search radius parameter, an increase in speed up factor also occurred along with an increase in main facility search radius. Where the increase in speed up factor was still not seen significantly when the radius was still below 40, while when the radius was more than 40 the increase in speed up factor started to increase significantly both in the parallel computing environment and computing cluster as shown in Figure 45.

![Figure 45 Speed up factor: Main facility search radius](image)

In the number of grid parameter, the increase in speed up factor was not too significant when the number of grid was still below 625, but when the number of grid was more than 625, the increase in speed up factor was increased significantly both in parallel computing environment and computing cluster environment as shown in Figure 26. It increased significantly both in parallel computing environment and computing cluster environment as shown in Figure 46.

![Figure 46 Speed up factor: Number of grid](image)
In the raw data size parameter, an increase in speed up factor occurs consistently in each particular raw data size in both the parallel computing environment and computing cluster environment as shown in Figure 47.

In facility dimension attribute parameter, speed up factor was inversely proportional to the increase in the number of dimension of the facility both in parallel computing and computing cluster environment. Although static speed up factor and even decrease, but the processing speed in cluster computing was still 45 times faster than in single computing in the number of dimension attribute as many as 8 and parallel computing is still 25 times faster than in single computing in the number of dimension attributes as many as 8 such as shown in Figure 48.
While on the parameter of distance between main facility and surrounding environment, there was no increase in speed up factor when distance parameter increase. Speed up factor tended to be static, even in environment cluster computing decreased at a distance of 50 to a distance of 100, but speed up factor in computing cluster environment was still 45 times faster than single computing and parallel computing environment and it was still 25 times faster than single computing. Although the speed up factor was static but the processing speed in both environments was still faster than the single computing as shown in Figure 49.

It can be concluded that the increase and decrease in speed up factors vary on each parameter, but the most prominent increase in speed up factor can be seen in the number of grid and raw data size parameters. The increase in speed up factor was different from the increase in processing time, although speed up factor looked static but in certain case processing time in cluster computing environment and parallel computing was still faster than single computing. Besides, the increase in the number of worker nodes in computing cluster did not add speed up the factors significantly compared to parallel computing despite a 15-fold increase in resources, it is because when distributing data to each cluster node, computing requires a longer communication time between nodes compared to parallel computing.

Mobile Application Interface

In this section, we showed the user interface of the application that has been built. There are three interfaces in form configuration interface, maps interface, and detail/information interface.

![Configuration Interface and Maps Interface](image-url)
The mobile device configuration interface, where the user can be configured the type of POI skyline to be searched as shown in Figure 50(a). User could also specify the parameter/preference of the POI skyline such as price level and rate level. Also, the user configuration interface could also determine the type of surrounding enumeration/facility along with the attribute of each of the surrounding facility that will be applied in the search for skyline object. After all the forms in the interface configuration are filled in, the user could press the Apply button to apply the set configuration while button change is used to clear all configuration forms if the user wants to change the setting of the skyline search. Figure 50(b) displayed maps interface, where this interface was the digital map displayed with the location center of the user’s presence and the skyline point coordinated is also calculated from the proposed algorithm. If the user click on the skyline point, the name of the skyline point will be displayed.

Figure 51 displayed the details of the skyline object obtained, where the detail interface displayed the coordination of the skyline object (longitude and latitude) and the detail of the values of the skyline object attributes specified. In addition, the detail interface also displayed the coordination of the surrounding environment (longitude and latitude) and the detail of the values of the attributes of the surrounding environment.

Advantage and Disadvantage

From the research conducted, it is realized that there is an advantage in using the Apache-spark to search spatial skyline based on the data streaming. It could be reduced computational time to enable data processing in real time. Whereas the disadvantages in this research were not having large data, and only using spatial data with real-time sources, while non-spatial data is still using
generated data. Also, the research only used one type of cluster computing framework, Apache-Spark, while there are many frameworks that can be tested on proposed algorithm and may have better performance. The proposed algorithm has not been able to calculate the speed of movement of the user towards the sliding window, the proposed algorithm has not been able to create a grid dynamically and proportionally based on the radius finding area.

5 CONCLUSION AND RECOMMENDATION

Conclusion

Apache Spark framework can increase the computational time of data streaming processing effectively and efficiently. Execution time shows that streaming data highly potential to be implemented in the proposed algorithm in real time. Thus, applying proposed algorithm for handling data streaming on an Android device in real time is achievable.

Recommendation

The next challenging work is to use larger data dimensions and non-spatial attributes with real-time sources, such as Traveloka, Zomato, or AiryRooms. Other cluster computing frameworks such as Apache Storm, Samza, and Flink can be considered to use. Challenges in the next research can include the test of movement speed as well as an algorithm that can determine grid size proportionally based on radius finding area dynamically.

REFERENCES


Appendix 1 Example of calculating spatial object distances using haversine formula

A user wants to know the distance that must be traveled from the city of San Francisco that has longitude ($\lambda_1$) and latitude ($\phi_1$) towards the city of Chicago that has longitude ($\lambda_2$) and latitude ($\phi_2$). Longitude and latitude values in each city as follows:

<table>
<thead>
<tr>
<th>City Name</th>
<th>Latitude ($\phi$)</th>
<th>Longitude ($\lambda$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>San Francisco</td>
<td>37,7749</td>
<td>122,4149</td>
</tr>
<tr>
<td>Chicago</td>
<td>41,8781</td>
<td>87,6289</td>
</tr>
</tbody>
</table>

It is known that the value of the earth’s surface radius is ($r = 6371$ km). Then the distance calculation between spatial objects uses the haversine formula as follows:

$$
\Delta \phi = \phi_2 - \phi_1
$$

$$
\Delta \phi = 41,8781 - 37,7749 = 4,1032
$$

$$
\Delta \lambda = \lambda_2 - \lambda_1
$$

$$
\Delta \lambda = 87,6289 - 122,4149 = -34,768
$$

$$
d = 2r \cdot \arcsin \left( \sqrt{\sin^2 \left( \frac{\Delta \phi}{2} \right) + \cos \phi_1 \cdot \cos \phi_2 \cdot \sin^2 \left( \frac{\Delta \lambda}{2} \right)} \right)
$$

$$
d = 2 \cdot 6.371 \cdot \arcsin \left( \sqrt{\sin^2 \left( \frac{4,1032}{2} \right) + \cos 37,7749 \cdot \cos 41,8781 \cdot \sin^2 \left( \frac{-34,768}{2} \right)} \right)
$$

$$
d = 2 \cdot \arcsin \left( \sqrt{0.78610 + 0.79042343 \cdot 0.74456675 \cdot 0.08937} \right)
$$

$$
d = 2 \cdot \arcsin(0.91580363)
$$

$$
d = 2 \cdot 6.371 \cdot 1,15750432
$$

$$
d = 14.748,9057733
$$

From the distance calculation using the haversine formula, the distance between a point in San Francisco testing a point in California is 14.748,9057733 km.
Appendix 2 Example of determine skyline object using SFS.

It is known that the user is looking for the closest hotel that has a low price. Where the user is at point (0,0), meanwhile the distribution of hotel coordinates at that location as follow:

![Diagram showing hotel locations and coordinates]

Meanwhile, the price and distance from the users in each of the hotels as follow:

<table>
<thead>
<tr>
<th>House</th>
<th>Price (in thousand €)</th>
<th>Distance (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>H1</td>
<td>100</td>
<td>1500</td>
</tr>
<tr>
<td>H2</td>
<td>1400</td>
<td>500</td>
</tr>
<tr>
<td>H3</td>
<td>700</td>
<td>600</td>
</tr>
<tr>
<td>H4</td>
<td>1300</td>
<td>1000</td>
</tr>
<tr>
<td>H5</td>
<td>900</td>
<td>1300</td>
</tr>
<tr>
<td>H6</td>
<td>1600</td>
<td>100</td>
</tr>
<tr>
<td>H7</td>
<td>400</td>
<td>300</td>
</tr>
<tr>
<td>H8</td>
<td>200</td>
<td>1200</td>
</tr>
<tr>
<td>H9</td>
<td>1000</td>
<td>200</td>
</tr>
<tr>
<td>H10</td>
<td>500</td>
<td>1400</td>
</tr>
<tr>
<td>H11</td>
<td>500</td>
<td>900</td>
</tr>
</tbody>
</table>

Furthermore, the price and distance values are normalized so that the calculation process can run better (in the example of normalization carried out by doing multiply 0.00001), so the price and distance for each hotel as follow:
Next, the entropy value is searched for each hotel using the following formula:

$$E_D(h_i) = \sum \ln(h_i + 1)$$

Where \( h_i \) is the value of non-spatial attributes in each object column. So that the entropy value of each hotel is like the one as follow:

<table>
<thead>
<tr>
<th>House ((h_i))</th>
<th>Price</th>
<th>Distance</th>
<th>( E_D(h_i) )</th>
<th># points that dominate</th>
</tr>
</thead>
<tbody>
<tr>
<td>H7</td>
<td>0.04</td>
<td>0.03</td>
<td>0.068779515</td>
<td>6</td>
</tr>
<tr>
<td>H9</td>
<td>0.1</td>
<td>0.02</td>
<td>0.115112807</td>
<td>2</td>
</tr>
<tr>
<td>H3</td>
<td>0.07</td>
<td>0.06</td>
<td>0.125927557</td>
<td>-</td>
</tr>
<tr>
<td>H8</td>
<td>0.02</td>
<td>0.12</td>
<td>0.133131313</td>
<td>2</td>
</tr>
<tr>
<td>H11</td>
<td>0.05</td>
<td>0.09</td>
<td>0.13496786</td>
<td>-</td>
</tr>
<tr>
<td>H1</td>
<td>0.01</td>
<td>0.15</td>
<td>0.149712273</td>
<td>0</td>
</tr>
<tr>
<td>H6</td>
<td>0.16</td>
<td>0.01</td>
<td>0.158370336</td>
<td>0</td>
</tr>
<tr>
<td>H2</td>
<td>0.14</td>
<td>0.05</td>
<td>0.179818427</td>
<td>-</td>
</tr>
<tr>
<td>H10</td>
<td>0.05</td>
<td>0.14</td>
<td>0.179818427</td>
<td>-</td>
</tr>
<tr>
<td>H5</td>
<td>0.09</td>
<td>0.13</td>
<td>0.208395329</td>
<td>-</td>
</tr>
<tr>
<td>H4</td>
<td>0.13</td>
<td>0.1</td>
<td>0.217527813</td>
<td>-</td>
</tr>
</tbody>
</table>

Furthermore, spatial objects are sorted by entropy in descending manner, and the skyline search process is carried out using fundamental skyline algorithms, where values dominated (having no better value than all attributes) by objects with the lowest entropy will be removed from the skyline candidate object. The skyline obtained from the example above is a hotel \( \{h_1, h_9, h_8, h_6\} \).
Appendix 3 Example of determine spatial skyline object using spatial skyline query based on surrounding environment

It is known that the distribution of objects in a location as follow:

Where the spatial object of the hotel is represented by \( h = \{h_1, h_2, h_3, h_4 \ldots, h_{20}\} \), while the spatial supermarket object is represented by \( s = \{s_1, s_2, s_3, h_4 \ldots, s_{20}\} \), meanwhile the restaurant is represented by \( r = \{r_1, r_2, r_3, r_4 \ldots, r_{20}\} \). Where the spatial attributes of each object as follow:

<table>
<thead>
<tr>
<th>ID</th>
<th>longitude</th>
<th>latitude</th>
<th>Facility Type</th>
<th>ID</th>
<th>longitude</th>
<th>latitude</th>
<th>Facility Type</th>
<th>ID</th>
<th>longitude</th>
<th>latitude</th>
<th>Facility Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>h_1</td>
<td>5</td>
<td>5</td>
<td>Hotel</td>
<td>s_1</td>
<td>2</td>
<td>4</td>
<td>Supermarket</td>
<td>r_1</td>
<td>3</td>
<td>4</td>
<td>Restaurant</td>
</tr>
<tr>
<td>h_2</td>
<td>8</td>
<td>11</td>
<td>Hotel</td>
<td>s_2</td>
<td>5</td>
<td>5</td>
<td>Supermarket</td>
<td>r_2</td>
<td>9</td>
<td>8</td>
<td>Restaurant</td>
</tr>
<tr>
<td>h_3</td>
<td>14</td>
<td>6</td>
<td>Hotel</td>
<td>s_3</td>
<td>5</td>
<td>12</td>
<td>Supermarket</td>
<td>r_3</td>
<td>6</td>
<td>1</td>
<td>Restaurant</td>
</tr>
<tr>
<td>h_4</td>
<td>7</td>
<td>7</td>
<td>Hotel</td>
<td>s_4</td>
<td>13</td>
<td>6</td>
<td>Supermarket</td>
<td>r_4</td>
<td>14</td>
<td>3</td>
<td>Restaurant</td>
</tr>
<tr>
<td>h_5</td>
<td>15</td>
<td>7</td>
<td>Hotel</td>
<td>s_5</td>
<td>3</td>
<td>2</td>
<td>Supermarket</td>
<td>r_5</td>
<td>7</td>
<td>5</td>
<td>Restaurant</td>
</tr>
<tr>
<td>h_6</td>
<td>1</td>
<td>8</td>
<td>Hotel</td>
<td>s_6</td>
<td>12</td>
<td>6</td>
<td>Supermarket</td>
<td>r_6</td>
<td>7</td>
<td>13</td>
<td>Restaurant</td>
</tr>
<tr>
<td>h_7</td>
<td>9</td>
<td>6</td>
<td>Hotel</td>
<td>s_7</td>
<td>8</td>
<td>3</td>
<td>Supermarket</td>
<td>r_7</td>
<td>12</td>
<td>15</td>
<td>Restaurant</td>
</tr>
<tr>
<td>h_8</td>
<td>4</td>
<td>2</td>
<td>Hotel</td>
<td>s_8</td>
<td>6</td>
<td>7</td>
<td>Supermarket</td>
<td>r_8</td>
<td>15</td>
<td>5</td>
<td>Restaurant</td>
</tr>
<tr>
<td>h_9</td>
<td>6</td>
<td>14</td>
<td>Hotel</td>
<td>s_9</td>
<td>9</td>
<td>7</td>
<td>Supermarket</td>
<td>r_9</td>
<td>12</td>
<td>4</td>
<td>Restaurant</td>
</tr>
<tr>
<td>h_10</td>
<td>13</td>
<td>8</td>
<td>Hotel</td>
<td>s_10</td>
<td>10</td>
<td>10</td>
<td>Supermarket</td>
<td>r_10</td>
<td>10</td>
<td>3</td>
<td>Restaurant</td>
</tr>
<tr>
<td>h_11</td>
<td>6</td>
<td>5</td>
<td>Hotel</td>
<td>s_11</td>
<td>8</td>
<td>6</td>
<td>Supermarket</td>
<td>r_11</td>
<td>14</td>
<td>2</td>
<td>Restaurant</td>
</tr>
<tr>
<td>h_12</td>
<td>12</td>
<td>2</td>
<td>Hotel</td>
<td>s_12</td>
<td>11</td>
<td>3</td>
<td>Supermarket</td>
<td>r_12</td>
<td>6</td>
<td>5</td>
<td>Restaurant</td>
</tr>
<tr>
<td>h_13</td>
<td>8</td>
<td>16</td>
<td>Hotel</td>
<td>s_13</td>
<td>9</td>
<td>2</td>
<td>Supermarket</td>
<td>r_13</td>
<td>13</td>
<td>5</td>
<td>Restaurant</td>
</tr>
<tr>
<td>h_14</td>
<td>15</td>
<td>2</td>
<td>Hotel</td>
<td>s_14</td>
<td>12</td>
<td>10</td>
<td>Supermarket</td>
<td>r_14</td>
<td>1</td>
<td>7</td>
<td>Restaurant</td>
</tr>
<tr>
<td>h_15</td>
<td>6</td>
<td>1</td>
<td>Hotel</td>
<td>s_15</td>
<td>1</td>
<td>2</td>
<td>Supermarket</td>
<td>r_15</td>
<td>9</td>
<td>12</td>
<td>Restaurant</td>
</tr>
<tr>
<td>h_16</td>
<td>8</td>
<td>10</td>
<td>Hotel</td>
<td>s_16</td>
<td>6</td>
<td>11</td>
<td>Supermarket</td>
<td>r_16</td>
<td>13</td>
<td>7</td>
<td>Restaurant</td>
</tr>
<tr>
<td>h_17</td>
<td>3</td>
<td>6</td>
<td>Hotel</td>
<td>s_17</td>
<td>5</td>
<td>4</td>
<td>Supermarket</td>
<td>r_17</td>
<td>4</td>
<td>4</td>
<td>Restaurant</td>
</tr>
<tr>
<td>h_18</td>
<td>10</td>
<td>15</td>
<td>Hotel</td>
<td>s_18</td>
<td>9</td>
<td>16</td>
<td>Supermarket</td>
<td>r_18</td>
<td>10</td>
<td>14</td>
<td>Restaurant</td>
</tr>
<tr>
<td>h_19</td>
<td>2</td>
<td>6</td>
<td>Hotel</td>
<td>s_19</td>
<td>13</td>
<td>3</td>
<td>Supermarket</td>
<td>r_19</td>
<td>5</td>
<td>8</td>
<td>Restaurant</td>
</tr>
<tr>
<td>h_20</td>
<td>16</td>
<td>8</td>
<td>Hotel</td>
<td>s_20</td>
<td>16</td>
<td>2</td>
<td>Supermarket</td>
<td>r_20</td>
<td>10</td>
<td>16</td>
<td>Restaurant</td>
</tr>
</tbody>
</table>

Next is a grid of a certain size, in this example the grid is built in size 2×2.
The next step is to find the main and surrounding facilities that have the best non-spatial attribute values as follow.

<table>
<thead>
<tr>
<th>Grid</th>
<th>Hotels</th>
<th>Restaurants</th>
<th>Supermarkets</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1_1</td>
<td>3</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>G2_1</td>
<td>2</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>G3_1</td>
<td>3</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>G4_1</td>
<td>6</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

From the calculation using the SFS method, a skyline object is obtained \{h_8, h_{15}, h_6, h_{17}, h_{19}, h_4, h_{11}\}. 
The author was born on 4th April, 1994 in Bogor, West Java. The author is the only child of Drs. H. Sopian, M.Pd and the late Dra. Hj. Ida Zubaedah. He entered State University of Jakarta in 2012 and graduated in 2016 with a degree in Bachelor of Vocational Education. His research interests lie in the area of software developing, ranging from design to implementation. After completing a bachelor's degree, he works at PT Electronic Data Interchange Indonesia (EDII) as a junior programmer. In 2017, he enrolled at the Bogor Agricultural University to get his master's degree in computer science. During his master's studies, he joined as a member in the Computational Intelligence and Optimization (CIO) laboratory, with a focus on his research in data mining field. For his master thesis, he’s working under the supervision of Dr Eng Ir Taufik Djatna, STP, MSi and Dr Eng Annisa, SKom, MKom. His thesis research is related to spatial data mining, skyline queries, and mobile application development.