khazanah informatika

Jurnal Ilmu Komputer dan Informatika

Location Selection Based on Surrounding Facilities in Google Maps using Sort Filter Skyline Algorithm

Annisa*, Salsa Khairina
Department of Computer Science
IPB University
Bogor, Indonesia
*annisa@apps.ipb.ac.id

Abstract-Selecting a good location is an essential task in many location-based applications. Intuitively, a place is better than another if there are many good facilities around it. The most popular location selection platform today is Google Maps. Unfortunately, Google Maps has not provided the location selection based on the number of surrounding facilities. Assume a situation when a college student wants to let a house near his campus. Besides the distance from the campus, the student certainly will consider amenities surrounding it, such as food courts, supermarkets, health clinics, and places of worship. The rent house will become a better choice if there are more of these facilities around. Skyline query is a well-known method to select interesting desirable objects. We applied the Sort Filter Skyline (SFS) Algorithm on Google Maps to get a small number of attractive locations based on the number of nearby facilities. This study has succeeded in developing a web-based application that facilitates Google Maps users to search for places based on the figure of surrounding facilities. The time required to do a location search using SFS in Google Maps will increase with the number of facility types considered by the user.

Keywords: location selection, skyline query, sort filter skyline, surrounding facilities

Article info: *submitted December 17, 2020, revised April 5, 2021, accepted April 21, 2021*

1. Introduction

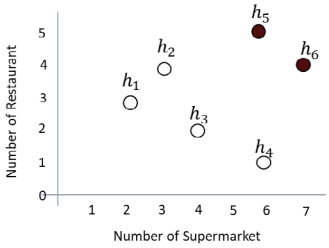
Selecting a good location is an important task in many location-based applications. Intuitively one location is better than another if there are many good facilities/ objects around it. Nowadays, there is an increasing need to take surrounding facilities into account when selecting a location. Chang et al. [1] explained when someone goes to a place for business or leisure, choosing the best hotel becomes very important. Syafrianto [2] and Popovic et al. [3] explained that hotel selection is strongly influenced by the goals and needs of visitors, not only in the form of hotel facilities, but also geographical surrounding and public facilities around the hotel. Another real world example is a situation when a college student wants to rent a house near his/her campus. Besides considering the distance from the rent house to the campus, the student certainly will also consider what facilities are available around the house, such as places to eat, supermarkets, health clinics, and also places of worship. The rent house will be considered as better option if there are more of these facilities around.

Skyline query [4] is a widely known method for selecting small number of interesting objects. Interesting objects, also known as skyline objects, are non-dominated objects in d-dimensional database. Borzsonyi et al. [4] defined that an object is said to dominate another object if it is equally good in all dimensions and better in at least one dimension. Figure 1 illustrates the skyline query problem. Consider a college student wants to rent a house. H1 to H6 are the houses for rent. Table in Figure 1 (a) shows the number of supermarkets and restaurants surrounding each house. Using skyline query algorithm, user can get the list of interesting rent houses based on the number of supermarkets and restaurants surround them. In Fig. 1 (b), H5 and H6 are skyline objects. H1, H2, H3, and H4 are dominated by H5 and H6, because H5 and H6 has more surrounding supermarkets and restaurants. H5 and H6 do not dominate each other because H5 has a greater number of surrounding restaurants but a lower number of surrounding supermarkets than H6 and vice versa. Using skyline query we can suggest H5 and H6 to the college student as options for renting a house. He/she can choose H5 if his/her preference is more surrounding restaurants, otherwise he/she can choose H6. There are many skyline

query algorithms, including [5, 6, 7, 8]. The skyline method has also been used in location and route selection such as in [9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20].

House	Number of Supermarket	Number of Restaurant
H1	2	3
H2	3	4
H3	4	2
H4	6	1
H5	6	5
H6	7	4

(a)



(b)

Figure 1. Illustration of skyline query problem

In [16], Arefin et al. uses skyline queries to address the problem of site selection by considering the type and number of surrounding facilities. Figure 2 shows an illustration of location selection problem in Arefin et al. [16]. Let us consider problem of the college student in finding rent house above.

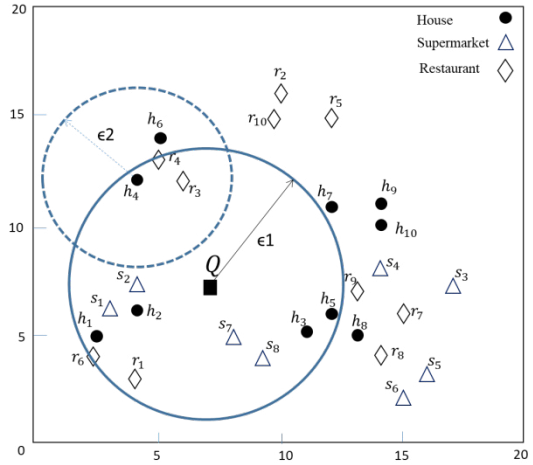


Figure 2. Illustration of location selection problem in Arefin et al. [15]

At first step, user gives the reference point (Q) and a radius distance (ϵ1). This Q reference point is generally a popular place in an area. In this example, campus is the reference point. Based on ϵ1 distance from Q, spatial objects of the target facility are selected. The houses (h) in Figure 2 are the examples of target facility. For the next step, we only consider rent houses within distance ϵ1 from Q. In the second step, the user determines the facility types that will be considered around the target (candidate) objects (houses within ϵ1 distance from Q). In the third step, if user considers n-surrounding facility types, then the number of facility types 1 to n at radius ϵ2 of each target object will be counted. In this example, user considers two types of facilities, restaurant (r)and supermarket (s). Thus the number of restaurant and supermarket within radius ϵ2 from each target objects is calculated and used as an attribute for each target object, to be combined with non-spatial information such as ratings, prices, and so on. Arefin et al. uses a variant of the R-tree index structure called aR-tree [21] to store spatial and non-spatial information from each facility types. In the last step, skyline query algorithm is performed like in Figure 1 to select skyline objects from the previous target objects. Arefin et al. applied Sort Filter Skyline (SFS), an algorithm introduced by Chomicki et al. [22] to find skyline objects. SFS is an enhancement of the naïve skyline algorithm, namely Block Nested Loops (BNL). SFS uses entropy function value to presort the dataset to reduce domination comparison.

Although the type of query like in [16] is very important in location selection application, unfortunately it has not been widely applied in our society. For this reason, this research aims to develop a web-based application so that users can run query in [16] on the most popular location selection platform, Google Maps. Many researches have been using Google Maps [23, 24, 25, 26, 27], however based on our knowledge, no research has been considering number and types of surrounding facilities for location selection in Google Maps.

Generally, Google Maps query, like Place Search and Nearby Search allow us to search for place information in specified area using a variety of categories. It returns a list of place location along with the summary information about each place, but never consider number of surrounding facilities for the selection. In this paper we developed a web-based application using SFS algorithm and Google Maps API so it can facilitate Google Maps users to search for location based on the number of surrounding facilities.

The remainder of this paper is organized as follows. In section II we briefly present location selection based on surrounding facilities problem in [16]. Section III provides our research methodology in detail. In section IV we conducted some experiments and explained them briefly in results and discussions. Finally, we put our conclusion in Section V.

2. Sort Filter Skyline (SFS) Algorithm

Skyline query has been widely used for location selection. Kodama et al. [28] and Wong et al. [29] introduced a framework for skyline query considering surrounding facilities using one type of facility. Arefin et al. [16] consider more types of facilities in location selection. In their research Arefin et al. explained that there are four calculation steps to obtain locations that take into consideration surrounding facilities according to user preference as we mentioned in the previous section. In the final step, the SFS algorithm is used to get skyline objects from a number of existing candidate objects.

Sort Filter Skyline (SFS) is a skyline query algorithm introduced by Chomicki et al. [22]. SFS is an enhancement of the naïve skyline algorithm, namely Block Nested Loops (BNL). Block Nested Loops (BNL) is an algorithm to read input data and compares each input with existing objects in memory. BNL does not use indexes and sorting. At first run, the first data directly enters the memory because there are no other objects in the memory. Subsequently

until all input data is read, if the input object is dominated by at least one object in memory then the input object is eliminated. If the input object is not dominated by one or more objects in memory, then the input object is nominated as skyline and any object that is dominated by the input object is removed from memory.

SFS is the development of the BNL algorithm by sorting data according to its entropy function values. Entropy formula is:

$$E(t) = \sum \ln(t[h_i]+1) \quad (1)$$

where is normalization of the candidate object attribute values (like rating and number of surrounding facilities). The entropy function above is always a monotone scoring function [8]. Based on Kalyvas et al. [30], the equation produces the most effective filtering of objects during skyline calculation. Intuitively, the smaller the entropy value of an object, the less likely it is to dominate.

Chomicki et al. [22] describes that an object in data which has a high entropy value can eliminate more objects away because it is guaranteed to dominate other objects. Therefore, processing sorted data has the advantage that no object in the data can be dominated by any object that enters afterwards. Thus, the data sorted by the value of the entropy function can eliminate the dominated object quickly.

The entropy function value is used to help filtering skyline objects by sorting data. SFS algorithm processes data that has been sorted using the entropy function. The object that has the largest entropy value is the skyline so it will go straight into memory because the object dominates other objects. After that, the next object is compared to the skyline object in memory. If the object is dominated by an object in memory then the object is eliminated because it is not a skyline object. On the other hand, if the object is not dominated by any object in memory then the object is saved to memory because it is a skyline object. Therefore, it is enough to compare an input object with skyline objects in memory without having to compare with all objects in the data. This happens because objects that have been previously eliminated are already dominated by skyline objects in memory, so it is enough to compare the input object with objects already in memory.

3. Methods

The data used in this research is Point of Interest (POI) data from Google Maps. POI data consists of spatial and non-spatial information. The spatial information is in the form of POI locations (latitude and longitude), while non-spatial information is in the form of POI type and POI rating. We use radius (near-by) feature that is already available in the Google Maps API to implement radius $\epsilon 1$ and $\epsilon 2$.

This study consisted of 5 stages. The first step is to determine the input, then proceed with making a data collection module. After the data obtained, then the data is processed by the Sort Filter Skyline (SFS) module, then the system is implemented, after that the system is tested, and finally conducting experiments with several scenarios. We used sample data to simplify the explanation of the method that we use in this research.

a. Determining Input Data

In this stage, we determined the input data needed to perform skyline operations. User input are spatial and non-spatial information. The information expected from the user are: the reference location that will become the reference point for searching spatial objects (Q), the type of object desired (type of target objects), the maximum radius from the reference location ($\epsilon 1$), the type of surrounding facilities, the maximum radius of the facilities from target objects ($\epsilon 2$), and the minimum rating from the surrounding facilities.

b. Creating Data Collection Module

The data collection module utilized two types of Google Maps' Place API. First we used Find Place to get geometry information (latitude and longitude) of the reference point, using reference point's place name as parameter. For example, if user input IPB University as the location of the reference point, the API returns geometry information of IPB University, which are latitude: -6.56636555 and longitude: 106.72148035. This geometry information is used as parameter to search for target objects surrounding the location of the reference point

Google Maps API Nearby Search is then used to get candidate objects and facilities surrounding the reference point, along with the required non-spatial information, which is rating. The parameters used are the geometry information of the search location point, the type of spatial object around the location of the reference point, the surrounding facilities (restaurants, ATMs, etc.), and the maximum radius. Table 1 shows candidate objects of place for rent within 0.7 km from IPB University: Landhius IPB Guest House, IPB International Dormitory, Amarilis Guest House, Al-Quds Boarding House, Arif Dormitory, and Dramaga Village Boarding House. After obtaining a candidate (target) object, this API is also used to find the number of facilities around the candidate object using the radius information from user. Maximum number of data obtained from requests for one type of POI is 60 data. User can input minimum rating of facility type, or can choose "None" in the system to not consider rating information from the facilities around the candidate object. This data collection module is run online to get data from Google Maps.

The results of this module are candidate objects data along with rating and number of facilities around them. Table 1 is an example of the results from the data collection module. Some candidate objects do not have rating info, so the rating value is displayed as 0.

Table 1. Example of data collection module results

Candidate Object	Rating	Restaurant-count	ATM-count
Landhius IPB Guest House	4.3	9	2
IPB International Dormitory	4.5	10	2
Wisma Amarilis	4.3	0	1
Arif Dormitory	0	9	1
Al-Quds Boarding House	0	9	2
Dramaga Village Boarding House	4.6	9	2

c. Creating a Sort Filter Skyline (SFS) Module

SFS algorithm is implemented using Python. Input data for SFS module is the result of the data collection module, as displayed in Table 1. Within SFS module, the input data is then sorted based on the highest to lowest entropy function values of the data calculated using (1), which results is presented in Table 2. Because the maximum value of the candidate object is 60, the candidate object attribute value is normalized by multiplying each candidate object by 0.001 so that the candidate object entropy value is between 0 and 1.

Table 2. Candidate location after sorted by entropy value

Candidate Object	Rating	Restaurant-count	ATM-count	Entropy Value
IPB International Dormitory	4.5	10	2	0. 016365
Dramaga Village Boarding House	4.6	9	2	0. 015480
Landhius IPB Guest House	4.3	9	2	0.015184
Al-Quds Boarding House	0	9	2	0.010940
Arif Dormitory	0	9	1	0.009950
Amarilis Guest House	4.3	0	1	0.005286

Subsequently, the objects that are dominated by other objects are removed. Thus, the obtained skyline objects are Dramaga Village Boarding House and IPB International Dormitory. Dramaga Village Boarding House is skyline because it has the highest rating, while IPB International Dormitory has the highest number of surrounding restaurants.

SFS algorithm is able to return skyline objects that consider several types of surrounding facilities from a candidate object. Currently, Google Maps is only able to consider just one type of facility from a location.

d. System Implementation

System implementation stage is carried out with the design and development of SSQ in web-based application. Web design phase is started by creating activity diagram that illustrates the flow of the system, as shown in Figure 3. Then simple mockups is made for developers to build the system.

Next, a web application prototype was developed based on activity diagrams and system requirements. The Google Maps API used is the same as in the data collection module, which is the Place API. The parameters required by the API are obtained from user input. Figure 4 and 5 is the result of the web interface on the location search feature based on the type and number of surrounding facilities which consist of the location search form page and the result recommendation location page.

The system processes Google Maps data to get location recommendations based on user input. Users are asked to fill in their preferences in a form, then the system provides a list of recommended locations according to the preferences given by the user.

Figure 4 is a form page that the user must complete. The reference location is a reference point for users to search for spatial objects. The type of object searched is the type of spatial object desired by the user around the reference point. The maximum object radius is the maximum radius from the reference point to find the desired object. Types of facilities around the object are the facilities preferred user wants around the spatial object. The maximum radius of the surrounding facility is the maximum radius of the facility of each object in unit of meter and the minimum rating of the surrounding facility is the minimum rating desired by the user for each facility type.

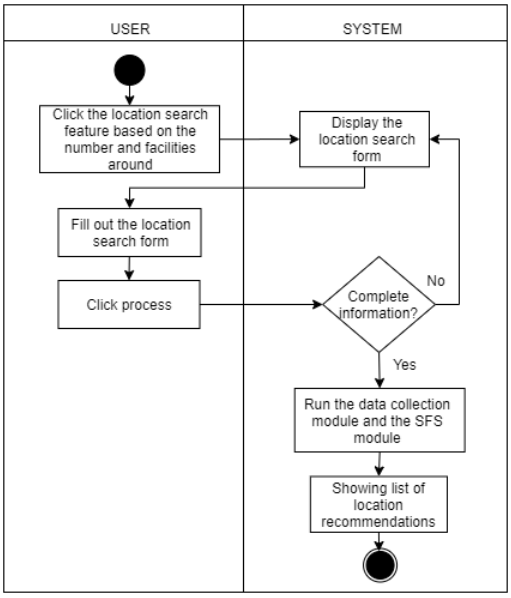


Figure 3. Activity Diagram

Figure 5 is a result page where there are red and green icons. The green icon is the location / reference point. The red icon is the result of the skyline which is the recommended locations for users based on the given preference.

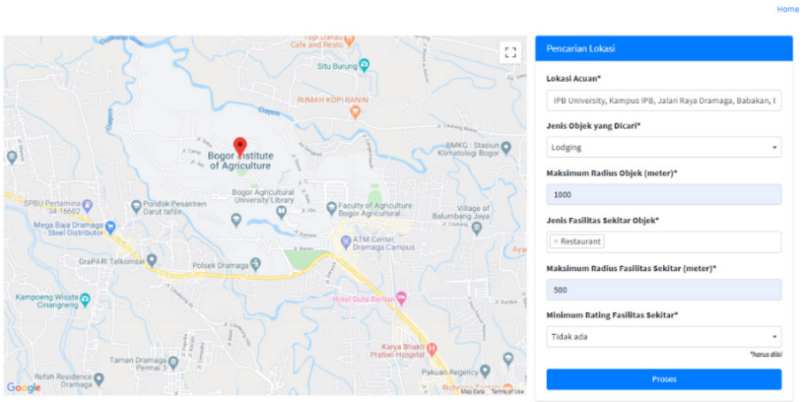


Figure 4. The web interface of form page

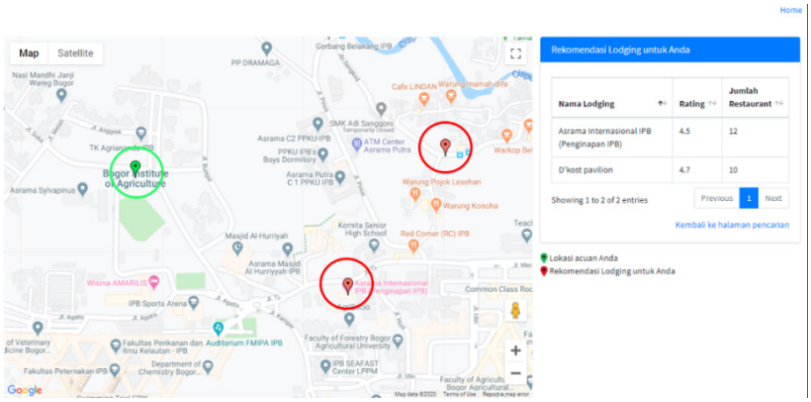


Figure 5. The web interface of result page

Figure 4 shows IPB University with green markers as the reference point to select interesting rent house with a maximum radius of 1 km. Facilities to be considered around rent house are restaurants with a maximum radius of 0.5 km without considering restaurant ratings. After processing, in Figure 5 shows the skyline rent house with red markers, IPB International Dormitory and D'kost Pavilion. Results form in Figure 5 also contain information about the number of restaurant around recommended skyline rent house along with the rating.

e. System Testing

This stage is carried out by testing the system that have been built using Blackbox testing method. Blackbox testing method is a software testing method that focuses on system functionality, specifically on the input and output without testing the algorithm. The test conducted was a system interaction from filling the form to producing a list of location recommendations based on input on the form.

f. Experiments

System performance tests are related to location search time on Google Maps. We conducted some experiments with these scenarios:

- Scenario 1: to test system performance towards the increase of number of surrounding facilities. We set the number of candidate objects to 5 objects, and increase the surrounding facilities by 2, 5, 10, 15, and 20.
- Scenario 2: to test system performance towards the increase of the radius (distance) of surrounding facilities from reference point Q. We fixed the number of candidate objects and reduce or enlarge the radius from reference point.
- Scenario 3: to test system performance towards the increase of number of surrounding facility type. Since the number of facility types is the number of dimension of candidate objects, we varies the number of facility type to 2D, 4D, 6D and 8D and varying the size of data for each facility from 5 to 20 data.
- Scenario 4: to test the effect of data collection module to the entire system performance.

3. Result and Discussion

a. System Testing

Blackbox testing method is used to test the results of the web that has been developed. Based on all the test results, it can be seen that all functions in the location search feature based on the type and number surrounding facilities have been successfully implemented. The implementation have included the form filling function to get a list of location recommendations based on input on the form. The results of the Blackbox test can be seen in Table 3 and code has been published in Github repository at https://github.com/salsakhairinaa/BasedSurroundingSkyline.git.

Table 3. Example of data collection module results

Testing Name	Testing Conditions	Test Result
Select the location search feature	If the user clicks on the location search feature based on the type and number of facilities around.	The Location Search form page appears.
Input the data	If all fields on the form have been filled in, then the user clicks the "Process" button.	The system will display a result page that lists the location recommendations along with rating information and the number of facilities to the user.
	If all fields on the form are not filled in, then the user clicks the "Process" button.	The system will remain on the form page and the message "Please complete the data" appears.

g. Experiments

Tests are performed to see the system performance measured in time required to complete the search. In the first scenario, the number of candidate objects is fixed at 5 objects, while the type of surrounding facilities is varied from 2, 5, 10, 15, to 20. The second scenario is to observe the effect of different radius on the system performance by setting two radius distance: a small radius (500 meters), and a large radius (5000 meters), while the number of candidate objects are fixed.

Each experiment was carried out with 10 iterations, and the result of each iteration is recorded. Table 4 shows the results of scenario 1 and scenario 2 tests. Figure 6 shows the results of the average execution time in seconds (d) which reveals that the execution time increases along with the increase in the number of facility type. The experiment also reveals that there is no big difference in the average execution time for small and large radius. This shows that the time required for location search based on surrounding facilities is greatly influenced by the number of facility types that are taken into account.

Figure 7 displays the experiment results from scenario 3. Figure 7 shows that the running time of the SFS algorithm increases with increasing data size. The execution time also increases with increasing data dimensions or the number of facility types around the candidate object. It can be seen that adding the data collection module in the system significantly increases processing time of location search using SFS algorithm.

Figure 8 shows experiment results of scenario 4, the execution time of the SFS algorithm which considers number of facilities. The data used in the SFS algorithm is set to 20 candidate objects with the number of facilities considered are 2, 4, 6, and 8. Figure 8 (a) is the execution time of the SFS algorithm in finding the location from existing data. Figure 8 (b) is the execution time starting from collecting data using Google Maps until determining the location using SFS. Based on the execution time, it can be concluded that the execution time increases with the increasing number of facilities being considered. The data used in this study are dynamic data so that the execution time in this study increases because of the data collection process, but this does not affect the performance of the SFS algorithm. The complexity of the SFS algorithm in the best case state is $O(dn + n\log n)$ and in the worst case is $O(dn^2)$, where d is the number of dimensions or number of facilities considered and n data size or number of candidate objects.

Table 4. Average of execution time and radius

Iterations	Number of Attributes					Radius	
	2	5	10	15	20	500	5000
1	11.54	48.46	96.64	116.47	203.33	14.42	14.51
2	10.06	45.18	95.38	184.06	157.59	13.60	13.33
3	9.40	45.39	83.63	125.23	205.47	14.27	14.73
4	10.70	55.01	97.71	103.20	204.02	14.12	15.49
5	10.11	57.44	98.26	105.44	199.63	13.32	14.84
6	13.77	53.48	93.99	115.34	201.49	14.44	13.27
7	13.74	55.49	94.90	113.29	197.62	12.02	16.47
8	9.49	44.51	88.48	110.68	201.98	12.72	15.87
9	10.91	56.27	88.73	118.45	199.93	14.31	13.51
10	9.02	56.08	87.98	111.14	201.09	13.75	14.02
Average of execution time (s)	10.87	51.37	92.57	120.33	197.21	13.70	14.60

4. Conclusion

This research has succeeded in implementing the SFS algorithm in Google Maps to answer location selection based on surrounding facilities query. The results of the study are a web-based application with simple user interface so that Google Maps users can run the query easily. The time required to search for a location depends on the number of facility types considered by the user. SFS algorithm performance is affected by the increase in data size and data dimensions.

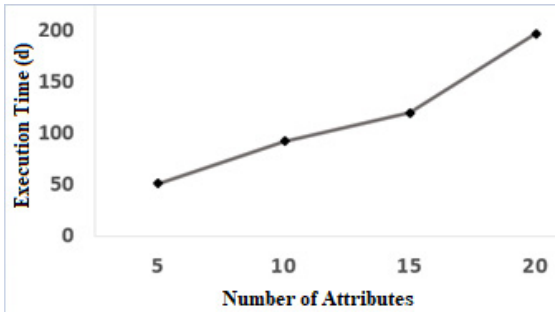


Figure 6. The average execution time based on increasing number of attributes

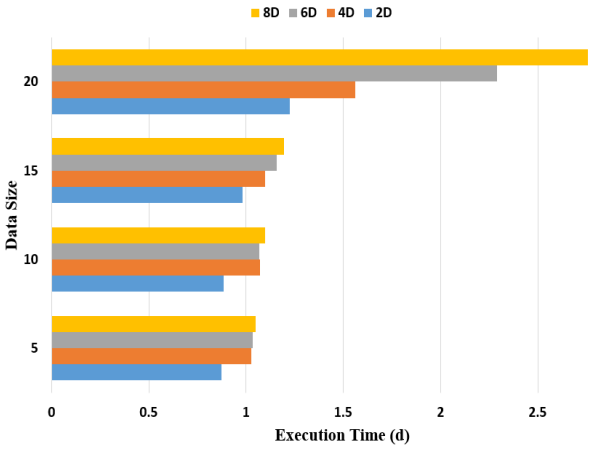
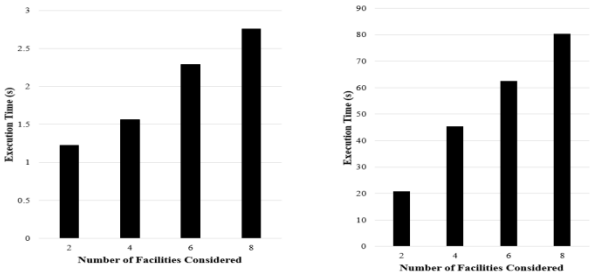


Figure 7. The average SFS execution time based on number of data and facilities



(a) **(b)**

Figure 8. (a) SFS algorithm execution time (b) The execution time of the data collection module is continued by the SFS module

Reference

[1] Z. Chang, M.S. Arefin, Y. Morimoto, Hotel recommendation based on surrounding environments, In Second IIAI International Conference on Advanced Applied Informatics, 2013, pp. 330-336.

[2] A. Syafrianto, "A Development of Spatial Skyline Query Based on Surrounding Environment for Data Streaming Using Apache-Spark", M.Kom. thesis, Computer Science, IPB University, Bogor, ID, 2010.

[3] G. Popovic, D. Stanujkic, M. Brzakovic, and D. Karabasevic, A multiple-criteria decision-making model for the selection of a hotel location. Land use policy, 2019, pp.49-58.

[4] S. Borzonyi, D. Kossmann, and K. Stocker, The skyline operator, In Proc. of ICDE, 2001, pp. 421-430.

[5] K.L. Tan, P.K. Eng, and B.C. Ooi, Efficient progressive skyline computation In Proc. of VLDB Conference, 2001, pp. 301-310.

[6] D. Kossmann, F. Ramsak, and S. Rost, Shooting stars in the sky: An online algorithm for skyline queries, In Proc. of VLDB Conference, 2002, pp. 275-286.

[7] D. Papadias, Y. Tao, G. Fu, and B. Seeger, An optimal and progressive algorithm for skyline queries, In Proc. of ACM SIGMOD Conference, 2003, pp. 467-478.

[8] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, Skyline with Presorting: Theory and Optimizations, In Proc. of the international IIS: IIPWM'06 conference, 2006, pp. 595-604.

[9] S. Shah S, A. Thakkar, S. Rami, A Survey paper on skyline query using recommendation system, In Journal of Data Mining & Emerging Technologies, 2016, pp. 1-6.

[10] M. Sharifzadeh, and C. Shahabi, The spatial skyline queries, In Proc. of VLDB, 2006, pp. 751-762.

[11] W. Son, M. Lee, H. Ahn, and S. Hwang, Spatial skyline queries: an efficient geometric algorithm, In Proc. of SSTD,2009, pp. 247-264.

[12] X. Guo, Y. Ishikawa, and Y. Gao, Direction-based spatial skylines, In Proc. of ACM SIGMOD Conference, 2010, pp. 73-80.

[13] K. Deng, X. Zhou, and H.T. Shen, Multi-source skyline query processing in road networks In Proc. of ICDE, 2007, pp. 796-805.

[14] M. Safar, D.E. Amin, and D. Taniar, Optimized

skyline queries on road networks using nearest neighbors, In Journal of Personal and Ubiquitous Computing, vol. 15, issue 8, 2011, pp. 845-856.

[15] Y.K. Huang, C.H. Chang, and C. Lee, Continuous distance-based skyline queries in road networks, In Journal of Information Systems, vol. 37, 2006. pp. 611-633.

[16] M.S. Arefin, Jinhao X, Zhiming C, Morimoto Y, Skyline query for selecting spatial objects by utilizing surrounding objects, In Journal of Computers, 2013, pp. 1742-1747.

[17] T. Djatna, F.H. Putra, dan A. Annisa, An Implementation of Area Skyline Query to Select Facilities Location Based on User's Preferred Surrounding Facilities. In Proc. of IEEE conference, ICACSIS, 2020, pp. 15-20.

[18] C. Li, A. Annisa, A. Zaman, M. Qaosar, S. Ahmed, and Y. Morimoto, Mapreduce algorithm for location recommendation by using area skyline query. In Algorithms, 11(12), 2018, pp.191.

[19] L.G. Asri, and A. Annisa, Application of Skyline Query on Route Selection (the Case Study of Bogor City Roadway). In the Proc. of IEEE conferences, International Conference on Computer Science and Its Application in Agriculture (ICOSICA), 2020, pp. 1-6.

[20] A. Annisa, A. Zaman, and Y. Morimoto, Area skyline query for selecting good locations in a map. Journal of Information Processing, 24(6), 2016, pp.946-955.

[21] D. Papadias, P. Kalnis, J. Zhang, and Y. Tao, Efficient OLAP operations in spatial data warehouses, In Lecture Notes in Computer Science, 2001, vol. 2121, pp. 443-459.

[22] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, Skyline with presorting, In Proc. of ICDE, 2003, pp. 717-816.

[23] E. Costa-Montenegro, F. J. González-Castaño, D. Conde-Lagoa, A. B. Barragáns-Martínez, P. S. Rodríguez-Hernández and F. Gil-Castiñeira, QR-Maps: An efficient tool for indoor user location based on QR-Codes and Google maps, In 2011 IEEE Consumer Communications and Networking Conference (CCNC), 2011, pp. 928-932.

[24] P. Pokorný, P., 2017, Determining Traffic Levels in Cities Using Google Maps. In Proc. of IEEE, The Fourth International Conference on Mathematics and Computers in Sciences and in Industry (MCSI), 2017, pp. 144-147.

[25] M.H. Erol and F. Bulut, Real-time application of travelling salesman problem using Google Maps API. In Proc. of IEEE, Electric Electronics, Computer Science, Biomedical Engineerings' Meeting (EBBT), 2017, pp. 1-5.

[26] C. Costa, J. Ha, and S. Lee, Spatial disparity of income-weighted accessibility in Brazilian Cities: Application of a Google Maps API. Journal of Transport Geography, *90*, 2021, p.102905.

[27] T. Listyorini and S. Muzid, Population resizing on fitness improvement genetic algorithm to optimize promotion visit route based on android and google maps API. In AIP Conference Proceedings, Vol. 1855, No. 1, 2017, p. 060001.

[28] K. Kodama, Y. Iijima, X. Guo, and Y. Ishikawa, Skyline queries based on user locations and preferences for making location-based recommendations, In Proc. of ACM LBSN, 2009, pp. 9-16.

[29] R.C. Wong, A.W. Fu, J. Pei, Y.S. Ho, T. Wong, and Y. Liu, Efficient skyline querying with variable user preferences on nominal attributes, In Proc. of VLDB, 2008, pp. 1032-1043.

[30] C. Kalyvas and T. Tzouramanis, A survey of skyline query processing, In arXiv preprint arXiv:1704.01788, 2017, pp. 19-20.