MapReduce-Based Computation of Area Skyline Query for Selecting Good Locations in a Map

Chen Li, Annisa, Asif Zaman, Yasuhiko Morimoto
Hiroshima University, Japan
Email: D165000@hiroshima-u.ac.jp, annisa@myfamily.web.id, asifzaman3180@yahoo.com, morimo@hiroshima-u.ac.jp

Abstract—Selection of good locations in a map is an indispensable function in many applications. In order to select specific locations, we have to specify detailed selection criteria. However, it is not easy especially for users of mobile devices. Therefore, we used an idea of skyline queries, which are known to be easy and effective to retrieve interesting data from a database. In our previous work, we have proposed area skyline query that selects good locations in a map. However, the query is not fast enough for handling “big data”. We simplify and revise the algorithm of the query in this paper by using MapReduce framework so that we can use it for big data. Experiments’ results demonstrate that the performance and scalability are superior to previous area skyline algorithm and are able to handle big data.

Index Terms—area skyline; grid structure; MapReduce;

I. INTRODUCTION

Many people are using GPS equipped mobile devices. To utilize the GPS function, there are many map applications for mobile devices. Selection of good locations is an indispensable function for such map applications. In general, we have to specify detailed selection criteria to select specific locations. However, it is not easy especially for users of mobile devices. Therefore, we used an idea of skyline queries, which are known to be easy and effective to retrieve interesting data from a database.

In general, a good location is close to train stations, shopping malls and sightseeing spots, etc. Furthermore, it should be far away from the facilities such as factories, noise sources, etc. Skyline query [4] is known to be effective to retrieve interesting data from a database. In our previous work [2], we have proposed area skyline query that selects good locations in a map.

Area Skyline Query

Assume a traveler would like to choose a destination which is closer to good touristic places and is far from unsafe places. In the situation, we can define dominance relationship between two places say $p_1$ and $p_2$ as follows:

Definition 1.1. (Area Dominance) We divide a map into $n \times n$ small square grids $G = \{g_{1,1}, \ldots, g_{n,n}\}$. A grid $g_i$ is said to dominate another grid $g_j$ in a map if distance to the nearest touristic place from $g_i$ is smaller than that of $g_j$ and distance to the nearest unsafe place from $g_i$ is larger than that of $g_j$.

Definition 1.2. (Area Skyline) Area Skyline is a set of grids, each of which is not dominated by another grid in a map.

Figure 1 is an example of a map. In the example, star marks, triangle marks, and square marks are places of facilities. We annotate “+” mark for the preferable facilities such as sightseeing spots for a tourist and “−” mark for the unpreferable facilities such as noisy factories. Star marks, denoted as $F_1^+ = \{f_{11}^+, f_{12}^+, f_{13}^+\}$, and triangle marks, denoted as $F_2^+ = \{f_{21}^+, f_{22}^+, f_{23}^+\}$, are preferable facilities and square marks, denoted as $F_3^- = \{f_{31}^-, f_{32}^-, f_{33}^-\}$, are unpreferable facilities. In the map, shaded grids are in the area skyline computed by the algorithm proposed in [2], which we call it “GASKY”. Each shaded grid is not dominated by another grid. On the other hand, each of white grids is dominated and we can eliminate such grids from candidates.

II. AREA SKYLINE ON MAPREDUCE

In this section, we propose a novel algorithm which is based on MapReduce framework, called MRGASKY, to reduce the response time of area skyline query. For simplicity, a map
is divided into $n \times n$ grids on average. We assume that the facilities inside the grids can be represented by the grids as the grids are small enough.

Our proposed algorithm consists of following two steps:

**Step1.** We compute the distance to the nearest facilities in same row:

**Step1.1** The Map function reads grids in the $i$-th row from left to right respectively to compute the distance, as illustrated in fig.2(a) (e.g., $f1_7$ in 2-th row of fig. 1). We assume that the distance of grids is infinite unless the first facility is encountered. And the distance of the grid is considered as 0 when the facility is encountered. Then the next grid is based on the previous grid plus one until the next facility is encountered.

**Step1.2** We select a minimum value of calculated two distances as final value of the distance, as illustrated in fig.2(b).

![Step1.1 Diagram](image1.png)

**Step2.** We compute Euclidean Distance to the nearest facility along with column wise in Reduce function:

**Step2.1** The Reduce function reads grids in same column based on the results of step 1(see fig.3). We map every $i$-th column to a two dimensional coordinate where $x$-axis represents the grids which are sorted from bottom to top, and $y$-axis represents the distance of step 1. The points are saved in a stack. It should be clear that each column has its own corresponding stack. We bisect the adjacent two points $p_i, p_j$, and name the intersection of the perpendicular bisector line $p_i p_j$ and x-axis as $x_{ij}$. If $x_{ij} > x_{jk}$ ($i < j < k$), the point $p_j$ then be deleted from stack (e.g., $x_{12} > x_{23}$ in fig.4, we delete $p_2$). $x_{ij}$ can be computed by formula (1):

$$x_{ij} = \frac{(y_j^2 - y_i^2) + (x_j^2 - x_i^2)}{2(x_j - x_i)}$$

where $(x_i, y_i)$ and $(x_j, y_j)$ are the coordinates of point $p_i$ and $p_j$.

![Step2.1 Diagram](image2.png)

**Step2.2** For every $i$-th column, we calculate the Euclidean Distance of deleted points (e.g., $p_1, p_2, p_4, p_5, p_6, p_8$ in fig.5). Specifically, we determine proximate intervals [11] of left points of step2.1. Fig.5 showed the intervals of shaded column in fig.3. $p_3$ and $p_7$ are two closest points to other grids in respective intervals in the same column. Then the Euclidean Distance of every grid can be calculated in the obvious way.

![Step2.2 Diagram](image3.png)

### III. EXPERIMENTS

In this section, we conducted 4 sets of experiments to compare the performance of GASKY and MRGASKY algorithms.
Experiments of GASKY is conducted on Linux operating system with Intel Core i7 3.40GHz processor with 4GB of RAM. And using this PC as one of four compute nodes. The other 3 nodes conduct on Linux operating system with Intel Core 2 3.16GHz and 2.13GHz processors, 4GB RAM. MRGASKY algorithm is implemented on Hadoop 2.5.2. In addition, we used synthetic datasets to evaluate our algorithm. Each experiment is repeated 10 times and we evaluated average processing time as the performance indicator. Since the step of removing dominated areas for both GASKY and MRGASKY algorithms is same, and the performance of this step is not different from other conventional skyline algorithms, we excluded it from the processing time calculation.

Effect on Grid Number

In these experiments, we used two sets of synthetic datasets, said DS_A1 and DS_A2. DS_A1 consists of 32 objects and 4 types of facilities which 2 types are preferable facilities and the other 2 are unpreferable facilities. We varied number of grids with $32 \times 32$, $64 \times 64$, $128 \times 128$, $512 \times 512$ and $1024 \times 1024$. For DS_A2, we fixed the number of facility types as 2 and fixed the number of objects as 2000. We varied the number of grids as $100 \times 100$, $500 \times 500$ and $1000 \times 1000$. We compared the effect on grid number in fig. 6 and fig. 7.

![Fig. 6. Processing time of DS_A1](image)

In fig.6, we can observe that the processing time of GASKY increases faster than MRGASKY when the number of grids is larger than $256 \times 256$. The reason is GASKY taking more time in $\min - \max$ computation when the number of grids increased. In Fig.7, when we raised the number of objects to 2000, we can observe that the processing time of GASKY increases faster than MRGASKY. The reason is that GASKY spent more time for building Voronoi Diagram and $\min - \max$ computation. Thus MRGASKY has better scalability than GASKY since grid number increasing.

Effect on Facility Types

For experiment DS_B in fig.8, we fixed the number of objects and grids as 10000 and $128 \times 128$. In addition, we varied the number of types to 2, 4, 6 and 8 respectively. And the number of preferable facility types is set to be same as the number of unpreferable facility types. Varying with the number of types, the processing time of GASKY increases linearly. The curve of MRGASKY is under GASKY and tends to stable. This result illustrated that performance of MRGASKY is also better than GASKY varying with facility types.

![Fig. 7. Processing time of DS_A2](image)

Effect on Object Number

For the effect on object number, the experiment DS_C in fig.9, we set the number of types as 2 and the grid size as $128 \times 128$. We raised the number of objects to 4000, 8000, 12000 and 16000. The results demonstrated that the processing time of the proposed algorithm is much smaller than previous works and and maintained stability enough to handling “big data”.

IV. RELATED WORKS

Skyline Query

Skyline operator is firstly proposed by Brozsonyi et al. [4]. Chomicki et al. proposed a skyline algorithm, Sort-Filter-Skyline (SFS), based on presorting BNL [6]. Tan at al. presented two progressive algorithms, Bitmap and Index, to improve performance of skyline computing [14]. A widely used effective algorithm nowadays, Brach and Bound Skyline (BBS), which is based on nearest neighbor search was developed by Papadias et al [15].
Spatial Skyline Query

Spatial Skyline Queries (SSQ) was firstly proposed by Sharifzadeh et al. [12]. Many literature proposed that the distance between objects and surrounding facilities should be considered as a parameter for selecting spatial objects [3], [7], [8], [9]. Based on previous work, You et al. proposed the farthest spatial skyline queries which is an efficient progressive algorithm, say Branch-and-Bound Farthest Spatial Skyline (BBFS), for exploiting spatial locality[16]. In [10], Lin et al. proposed EFFN algorithm to firstly considered unfavorable facilities. Different from above literature, our previous work, Unfixed-Shape Areas Skyline (UASKY), which divided the target area into several disjoint subareas by using Voronoi Diagram. The subareas may be further divided to sub-subareas by other facility types. For each sub-subareas, they calculated the maximum and minimum distance to closest facility for each $F^+\text{ and } F^-\text{ in}[1]$. In [2], we proposed a outperformed algorithm to UASKY, Grid-based Area Skyline (GASKY), which divided the target area into several grids on average. Then they used Voronoi Diagram to divide the grids, and calculated maximum and minimum distance for selecting good locations.

MapReduce Based Query Processing

Since the cost of spatial skyline queries is higher than conventional skyline queries, Zhang et al. designed three MapReduce based BNL (MR-BNL), MapReduce based SFS (MR-SFS) and MapReduce based Bitmap (MR-Bitmap) algorithms for processing skyline queries [17]. Chen et al. applied an angular data partition in the MapReduce-based solution for skyline query evaluation [5]. In [13], Siddique et al. proposed k-dominant skyline query computation in MapReduce Environment. However, few papers proposed solution on spatial skyline queries. This motivated us to propose an effective algorithm for selecting good locations by using MapReduce.

V. CONCLUSIONS

Selection of good locations which are close to preferable facilities and far from unfavorable facilities are important for various applications. GASKY can help to find such areas which are not dominated by another areas. Our proposed MRGASKY is using MapReduce to implement GASKY which has a better performance and scalability when increase grid number, facility types and object number. In future, we will consider about $k-$dominant skyline areas problem, which is a variant of skyline query for a map. In addition, we will consider effective utilization of non-spatial properties such as price and population density, as an addition parameters in our proposed selection algorithm.

ACKNOWLEDGMENTS

This work is supported by KAKENHI (16K00155,17H01823) Japan. A. Zaman are supported by Japanese Government MEXT Scholarship. Annisa is supported by Indonesian Government DG-RSTHE Scholarship.

REFERENCES

[1] Annisa, Siddique, M.A., Zaman, A. and Morimoto, Y.: A Method for Selecting Desirable Unfixed Shape Areas from Integrated Geographic Information System, Proc. IAI 2015, pp.195–200 (2015).
[2] Annisa, A., Zaman, Y.: Morimoto: Area skyline query for selecting good locations in a map, J. Inf. Process, 24(6), 946055 (2016).
[3] Arefin, M., Xu, J., Chen, Z. and Morimoto, Y.: Skyline Query for Selecting Spatial Objects by Utilizing Surrounding Objects, Journal of Software, Vol. 8, No. 7, pp. 1742–1749 (2013).
[4] Borzsonyi, S., Kossmann, D. and Stocker, K.: The skyline operator, Proc. 17th International Conference on Data Engineering (ICDE), April 2-6, Heidelberg, Germany, pp.421-430 (2001).
[5] Chen, L., Hwang, K., and Wu, J.: MapReduce skyline query processing with new angular partitioning approach. InPDPS Workshops, pp. 2262–2270 (2012).
[6] Chomicki, J., Godfrey, P., Gryz, J. and Liang, D.: Skyline with Presorting, Proc. 19th International Conference on Data Engineering (ICDE), March 5-8, Bangalore, India, pp. 717-719 (2003).
[7] Guo, X., Ishikawa, Y. and Gao, Y.: Direction-based Spatial Skylines, Proc. 9th ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE), June 6, Indiana, USA, pp.73–80 (2010).
[8] Kodama, K., Iijima, Y., Guo, X. and Ishikawa, Y.: Skyline Queries Based on User Locations and Preferences for Making Location-based Recommendations, Proc. 19th International Workshop on Location Based Social Networks (LBSN), November 3, Washington, USA, pp.9–16 (2009).
[9] Lin, Q., Zhang, Y., Zhang, W. and Lin, X.: Efficient general spatial skyline computation, World Wide Web, Vol. 16, No. 3, pp. 247–270 (2013).
[10] Lin, Y.-W., Wang, E.-T., Chiang, C.-F. and Chen, A.L.P.: Finding Targets with the Nearest Favor Neighbor and Farthest Disfavor Neighbor by a Skyline Query, Proc. 29th Annual ACM Symposium on Applied Computing (SAC), March 24–28, Gyongju, Korea, pp.821–826 (2014).
[11] Man, D., Uda, K., Ito, Y., and Nakano, K.: Accelerating computation of Euclidean distance Map using the GPU with efficient memory access, The International Journal of Parallel, Emergent and Distributed Systems, June 12, pp.1-23 (2012).
[12] Sharifzadeh, M. and Shahabi, C.: The Spatial Skyline Queries, Proc. 32nd International Conference on Very Large Data Bases (VLDB), September 12–15, Seoul, Korea, pp.751–762 (2006).
[13] Siddique, M. A., Tian, H., Morimoto, Y.: k-dominant skyline query computation in MapReduce environment. IEICE Trans. Inf. Syst. 98, 17451361 (2015).
[14] Tan, K.-L., Eng, P.-K. and Ooi, B.C.: Efficient Progressive Skyline Computation, Proc. 27th International Conference on Very Large Data Bases (VLDB), September 11–14, Rome, Italy, pp.301–310 (2001).
[15] Xia, T., Zhang, D. and Tao, Y.: ON Skylining with Flexible Dominance Relation, Proc. 24th International Conference on Data Engineering (ICDE), April 7-12, Cancun, Mexico, pp. 1397–1399 (2008).
[16] You, G.-W., Lee, M.-W., Im, H. and Hwang, S.-W.: The Farthest Spatial Skyline Queries, Information Systems, Vol. 38, No. 3, pp.286–301 (2013).
[17] Zhang, C., Zhou, S., and Guan, J.: Adapting skyline computation to the MapReduce framework: Algorithms and experiments. In DASFAA Workshops, pp. 403–414 (2011).