

An Effective Filtering Method for Skyline Queries in MANETs

Mi-Ra Park, Min-Kee Kim, Jun-Ki Min

Dept of Information Media Engineering, Korea University of Technology and Education, Korea
{happypmr, kintelk, jkmin}@kut.ac.kr

Abstract— In this paper, we propose an effective filtering method for skyline queries in mobile ad hoc networks (MANETs). Skyline query processing is important since skyline objects show interesting points from a large data set according to multiple criteria. Most skyline researches about wireless ad hoc network focus on optimizing the energy consumption due to the limited resource. But, most existing researches assume that data is uniformly distributed and do not consider data whose distribution is skewed in a specific region.

In order to reduce the energy consumption, we propose a new filtering method considering the data distribution. We verify the performance of the proposed method through a comparative experiment with an existing method. The results of the experiment confirm that the proposed method reduces the communication overhead and execution time compared to the existing method.

Keywords— P2P, MANET, skyline, skewed data, filtering

I. INTRODUCTION

According to the various expansions of wireless peer-to-peer (P2P) techniques, searching problems for distributed databases into P2P communication have been gaining increasing interest in mobile ad hoc networks (MANETs). MANETs are a collection of mobile nodes that dynamically self organize in a wireless network without using any pre-existing infrastructure [1]. MANETs primarily aim to use military communication networks but become available on general commercial networks, such as sensor networks or home networks in recent years. In this paper, we consider wireless communications between handsets without the presence of central servers [6]. The data stored is assumed to be distributed on handsets.

A handset has limited bandwidth and energy constraints in MANETs. The originator of a query needs to obtain data from surrounding mobiles. At this point, we have to reduce the amount of data to be transferred because of energy constraints.

Assuming a relational setting, a skyline query returns a set of tuples that are not dominated by others [3]. A tuple p is said to *dominate* another tuple q , if p has equal or smaller values than q in all dimensions, and has a smaller value in at least one dimension. Figure 1 shows the skylines of hotels

that have cheap and high quality rating. $h1$ cheaper than $h2$ and the quality of $h1$ is better than that of $h2$. So, $h1$ dominates $h2$. And, since there is no hotel dominating $h1$, $h1$ is a skyline object.

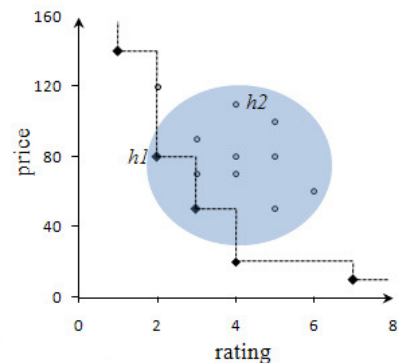


Figure 1. A example of the skewed data

Most previous works on skyline queries [3, 4, 5, 6, 9] assume that data is uniformly distributed. Our work considers skyline query processing in a mobile context with the following assumptions: 1) Data does not follow a uniform distribution. As shown in Figure 1, hotel objects severely appear in the region represented a circle. 2) Frequently occurring data on a device M_1 tends to frequently occur on a device M_2 . 3) Mobile users posing skyline queries are only interested in data pertaining to a limited geographical area [6].

In this paper, we propose a method called *Filtering Tuple using a Mode value* (i.e., the most frequent value) to reduce the communication overhead in MANETs. To find a mode value, we measure the frequencies of values on each dimension and choose the filtering tuple based on the mode values. Also, we show that the performance of our proposed filtering technique compared to that of an existing filtering technique.

The rest of this paper is organized as follows. Section 2 reviews the existing work on computing the skyline and skyline queries on mobile devices. Section 3 presents the method of selecting the filtering tuple using the mode values. In Section 4, we explain the details of our technique in mobile communication environments. Section 5 shows our experimental results. And Section 6 concludes this paper.

II. RELATED WORK

Our work is motivated by previous studies on both skyline query processing and skyline queries on mobile devices in MANETs.

A. Skyline Query Processing

Borzsonyi *et al.* [3] introduced first study for the skyline computation. It proposed skyline query processing through *Block Nested Loop* (BNL) algorithm and *Divide and Conquer* (D&C) algorithm. The BNL algorithm repeatedly scans a set of records, and compares data stored in the memory. The D&C algorithm recursively divides the objects into several partitions, then produce skyline objects in each partition. Then, it produces the final skyline objects by merging the local skyline in each partition. Later, Chomicki *et al.* [4] proposed *Sort Filter Skyline* (SFS) algorithm as a variant of BNL. It makes every attribute single value by operating, and reduces comparable objects by sorting.

Apart from skyline queries in the centralized pattern, Balke *et al.* [5] addressed distributed skyline query processing in for web information systems. Sun *et al.* [8] addressed the problem of processing skyline queries on multiple tables in a distributed environment. Cui *et al.* [9] proposed *PadSkyline* (Parallel Distributed Skyline query processing) algorithm based on the *reduced minimum bounding box* (*rMBR_i*). Yoon *et al.* [10] proposed Distributed Spatial Skyline (DSS) algorithm to compute distributed spatial skylines in wireless sensor networks.

Recently, processing skyline queries on P2P and ad hoc networks has attracted much attention. Wang *et al.* [7] addressed skyline query processing on peer-to-peer networks. Antony *et al.* [11] proposed to use skyline queries over ad hoc aggregations. Like related literatures, in this paper, we consider combined environments of ad hoc and P2P technologies.

B. Skyline Queries on Mobile Devices

Huang *et al.* [6] proposed a technique for skyline query processing on mobile devices in MANETs. They proposed a filtering method and the selection criteria for the filtering tuple in MANETs.

All local skyline objects of each device are not contained in final skyline set. Thus, to reduce the communication overhead, potential objects to be the final skyline are only transmitted to the query originator rather than transmitting all local skyline objects of each device. A filtering method prevents unnecessary transmission of data based on a tuple, called the filtering tuple.

In other words, when a user poses a query to a mobile device (i.e., query originator) M_{org} , M_{org} obtains skyline objects from its local data. And M_{org} sends the filtering tuple T_{org} , which is a skyline object in local skyline objects, to an adjacent device M_I . M_I also obtains its local skyline objects. Then, some local skyline objects of M_I which are dominated by T_{org} are pruned out since these objects cannot be final skyline objects. Thus, the communication overhead to transmit local skyline objects is reduced.

M_I also selects its filtering tuple T_I among remained local skyline objects and T_{org} based on the selection criteria for the filtering tuple and sends T_I to other nodes. Following these steps, skyline objects in MANET are computed.

A selection of the filtering tuple is important to reduce the communication overhead since the volume of pruned local skyline objects changes according to the filtering tuple.

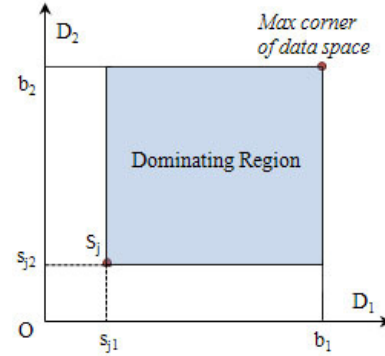


Figure 2. Example of Dominating Region

In [6], *VDR* (Volume of Dominating) is used as the selection criteria. Figure 2 illustrates the concept of *VDR*. Suppose that the range on a dimension D_k is $[0, b_k]$ in the virtual global relation R . Then the volume of a tuple S_j 's domination region is $VDR_j = \prod_{k=1}^d (b_k - s_{jk})$. As shown in Figure 2, an object in the dominating region, represented by a gray box, is dominated by S_j . Therefore, as increases the volume of S_j 's dominating region, increases the number of objects dominated by S_j under the assumption of uniform distribution. Consequently, a tuple from skyline objects with the maximum *VDR* value is selected as the filtering tuple.

The filtering tuple is dynamically updated during the procedure of query relay to increase its pruning potential. After doing a local skyline query on other mobile device, if there is a tuple with larger maximum *VDR* then the previous filtering tuple, the filtering tuple is changed.

But, in the skewed distribution, *VDR* does not work effectively. Thus, we propose an effective filtering method considering the skewed distribution.

III. FILTERING TUPLE ON MODE

In this paper, we introduce the efficient skyline query processing by filtering in each single mobile device.

A. Problem Definition

We assume a setting with n mobile devices $M = \{M_1, M_2, \dots, M_n\}$. Each device M_i holds a database relation R_i .

As shown in Table 1, a mobile device M_I holds hotel relations R_I . Assume that a user wants the information on cheap and good hotels. Each hotel tuple has attributes for the price and the rating based on service quality.

The skyline objects on M_I are $\{h_1, h_6, h_{12}, h_{14}, h_{15}\}$. Assume that the maximum values of price and rating are 200 and 10, respectively.

Table 1. Relation R_I

hotel	price	rating	hotel	price	rating
h_1	140	1	h_9	70	3
h_2	120	2	h_{10}	70	4
h_3	110	4	h_{11}	60	6
h_4	110	5	h_{12}	50	3
h_5	90	3	h_{13}	50	5
h_6	80	2	h_{14}	20	4
h_7	80	4	h_{15}	10	7
h_8	80	5			

We need to pick the filtering tuple from M_I 's local skyline. Using the VDR definition in Section 2.B, we have $VDR_1 = (200 - 140) * (10 - 1) = 540$, $VDR_6 = (200 - 80) * (10 - 2) = 960$, $VDR_{12} = (200 - 50) * (10 - 3) = 1050$, $VDR_{14} = (200 - 20) * (10 - 4) = 1080$, $VDR_{15} = (200 - 10) * (10 - 7) = 570$. Because h_{14} has the largest VDR value, h_{14} is selected as the filtering tuple.

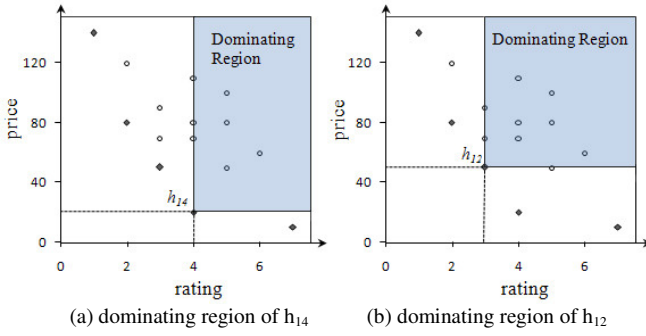


Figure 3. Example of Dominating Region

Figure 3 presents the dominating regions of h_{12} and h_{14} , respectively. The dominating region of h_{14} is larger than the dominating region of h_{12} . But, h_{14} dominates seven objects, whereas h_{12} dominates nine objects. Thus, VDR does not precisely reflect the number of dominated objects. This discrepancy appears severely, as the skewness of data is high.

B. A Mode Value of Data

There are many works to analyse the skewness of the data distribution. But since, in MANETs, the computing power of each node is restricted, a complex technique cannot be applied. Thus, we use the mode values to improve efficiency of the filtering tuple under the skewed distribution.

The basic idea of our filtering technique is to choose the filtering tuple whose dominating region covers the mode values in all dimensions since the objects appear around the mode value on each dimension. When each device processes a skyline query, frequencies of values on all dimensions can be obtained. Thus, it has no additional cost. Then, the mode values of all dimensions are obtained using the frequencies of values. Actually, we do not count the frequency of individual value due to the memory and bandwidth restriction. Instead, we use equi-width histogram. In our experiment, the number of intervals on a dimension is ten.

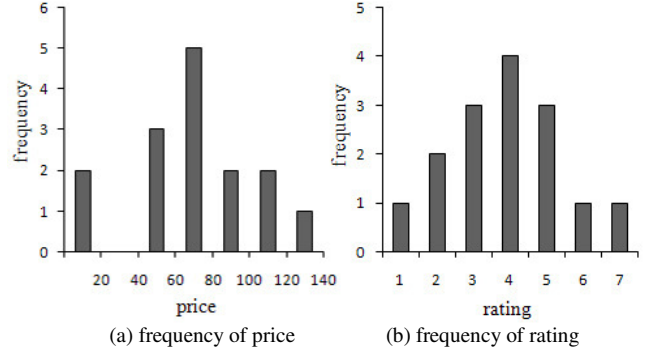


Figure 4. Frequency of value

Figure 4 shows frequency of values in Table 1. In this example, the number of interval on each dimension is ten but only present seven due to space restriction. The mode value of price is in 60-80 block. Its average is 70. The mode value of rating is 4. We use MP (Mode Point) to present mode values of each dimension. MP is (70, 4) in this example.

C. Choosing Strategy for Filtering tuple

The filtering tuple is used to reduce the transmission cost by pruning skyline objects having no possibility to be a final skyline object.

In this section, we present how to select the filtering tuple using MP . We use a 2-D illustration as shown in Figure 5 for our explanation. Basically, the closest skyline object to a line passing through O and MP is the filtering tuple.

First, we draw a line that passes through O and MP . Then, we compute the distance between a skyline object and the straight line. The distance between a point and a line is calculated using inner product of vector space and Pythagorean Theorem.

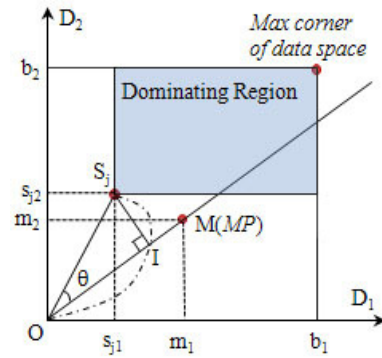


Figure 5. Example for DPL computation

Suppose that the value range on a dimension D_k is $[0, b_k]$. M presents the mode point MP . And, S_j is a skyline object.

The formula for the distance DPL_j from a point S_j to the line $\overrightarrow{OM}$ is the follow:

$$DPL_j^2 = |\overrightarrow{OS_j}|^2 - |\overrightarrow{OI}|^2$$

$$\begin{aligned}
&= |\overline{OS_j}|^2 - |\overline{OS_j}|^2 \cdot (\cos\theta)^2 \\
&= \sum_{k=1}^d \left(\frac{S_{jk}}{b_k} \right)^2 - \frac{\left(\sum_{k=1}^d \frac{m_k \cdot S_{jk}}{b_k^2} \right)^2}{\sum_{k=1}^d \left(\frac{m_k}{b_k} \right)^2} \quad (1)
\end{aligned}$$

, where d is the number of dimensions.

First of all, we consider the normalized distance in order to assign equal weight in each dimension. Thus, the value in each dimension D_k is divided by the upper bound b_k .

We calculate distance $|\overline{OS_j}|$ between point O and point S_j . And, we can calculate $\cos\theta$ using $\overline{OS_j}$, $\overline{OM}$. Then, we calculate $|\overline{OI}|$ using $|\overline{OS_j}|$ and $\cos\theta$. Since both the length of sides $|\overline{OS_j}|$ and $|\overline{OI}|$, we calculate the length of $|\overline{IS_j}|$ using Pythagorean Theorem. With some algebra, Eq (1) is derived as the final DPL formula.

A mobile device computes DPL of each skyline object. The filtering tuple is a tuple that has the minimum value of DPL . For example, in Table 1, we choose h_{12} as the filtering tuple that has the minimum value of DPL .

IV. ADAPTATION TO MANET

A mobile device has a part of information that a user wants to retrieve due to the memory restriction. So, we assume that the query originator communicates with several mobile devices based on multiple hop routing. A user is interested in not only data in a query originator but also data pertaining to other devices.

A user executes a skyline query. First, the query originator M_{org} computes the local skyline objects. During this time, M_{org} computes frequencies of values in each dimension. Then, M_{org} can find the mode values using the frequencies. The mode point MP described in Section 3.B consists of mode values in all dimensions. M_{org} selects a local skyline object as the filtering tuple that has the smallest value of DPL . The query originator M_{org} sends the query with the filtering tuple and frequencies of values. At the end of the process, M_{org} merges results of other devices and generates final skyline objects.

An adjacent device M_i receives the query request with the filtering tuple and the frequencies. M_i compares the filtering tuple with local objects. Since a local object dominated by the filtering tuple cannot be a skyline object, such objects are not involved in the local skyline processing. Through this process, we reduce the execution time by reducing the number of local objects to be computed. During this time, M_i accumulates the frequencies transmitted from device M_{org} . Then, M_i processes the skyline query using the remaining objects.

The device M_i finds a new MP and selects a new filtering tuple among local skyline objects and the filtering tuple transmitted from M_{org} .

Algorithm1 originator_skyline()

Input

S : dataset of local

Output

Z : final skyline

Procedure

```

1:  $F, SK_i$  // frequency of data, local skyline set
2: for each tuple  $S_j$  in  $S$ 
3:    $F = \text{updateFrequency}(S_j, F)$ 
4:   if ( $S_j$  is skyline) then // compute skyline
5:     insert  $S_j$  into  $SK_{org}$ 
6:   end for
7: for each  $f$  of  $F$ 
8:    $MP \leftarrow \text{mode of } f$  // find a mode point
9: end for
10: for each tuple  $S_j$  in  $SK_{org}$ 
11:    $T \leftarrow \text{minDPL}(S_j)$ 
12: end for
13:  $SK_1, SK_2, \dots, SK_n \leftarrow \text{surrounding\_skyline}(T, F)$ 
14:  $Z = \bigcup_{i=1}^n SK_i$ 
15:  $Z = Z \cup SK_{org}$ 
16: return  $Z$ 

```

Figure 6. Local Skyline Query on query originator

Figure 6 presents an algorithm, `orginate_skyline()` for the local skyline query processing on the query originator M_{org} . We briefly explain the routine and variables. S is a dataset in M_{org} . The routine `minDPL( $S$ )` returns a tuple that has the minimum value of DPL among local skyline objects. `updateFrequency( $T, F$ )` is a function to update frequencies. F is object that will input frequency of data. SK_i is a local skyline set of a mobile device M_i .

First, it scan S to find a skyline set. We update F during scanning. The skyline objects inserted into SK_{org} (lines 2-6).

We compute the mode value of each dimension using frequency stored in F , and make MP (lines 7-9). Then, we compute DPL value of each skyline object. And we choose a object which has the minimum value as the filtering tuple (lines 10-12). The query originator sends around other devices the query, the filtering tuple and F .

When other devices M_j s return sets of local skyline objects SK_j s, SK_j s and SK_{org} are merged into Z while also removing the non-qualifying objects. (lines 13-16). Then a user receives the final skyline result Z .

In Figure 7, `surrounding_skyline()` shows the procedure of processing the skyline using the filtering tuple in a device. A device M_j receives T and F from another device. First, M_j scans its own dataset S and prunes the tuples that dominated by T . M_j accumulates the frequencies F during this step (lines 1-5). Then, M_j computes skyline of remaining data (lines 6-9). We find a new mode point using accumulated F (lines 10-12). And, M_j computes DPL of each local skyline object.

And a tuple, which has the smallest DPL values among local skyline objects and the filtering tuple received from the previous device, is selected as the new filtering tuples (Line 13-16). Finally, we return SK to query originator (line 17).

Algorithm2 surrounding_skyline(T, F)

Input

S : local dataset
 T : the filtering tuple
 F : frequency of data of previous devices.

Output

SK : local skyline set

Procedure

```
1: for each tuple  $S_j$  in  $S$ 
2:    $F = \text{updateFrequency}(S_j, F)$ 
3:   if (  $T$  dominate  $S_j$  ) then // filtering course
4:     remove  $S_j$  from  $S$ 
5:   end for
6: for each  $S_j$  in  $S$ 
7:   if (  $S_j$  is skyline ) then // compute skyline
8:     insert  $S_j$  into  $SK$ 
9:   end for
10: for each  $f$  of  $F$ 
11:    $MP \leftarrow \text{mode of } f$  // find a mode point
12: end for
13: for each tuple  $S_j$  in  $SK$ 
14:   if (  $DPL$  of  $S_j < DPL$  of  $T$  ) then
15:      $T \leftarrow S_j$  // more efficient filtering tuple
16:   end for
17: return  $SK$ 
```

Figure 7. Local Skyline Query on surrounding device

V. EXPERIMENT EVALUATION

In this section, we proceed to offer insight into the properties of the method proposed in this paper based on experiment. We study the efficiency of the filtering tuple in local skyline computation in a simulated MANET environment.

The experiments is conducted on an Inter® Core™2 Duo CPU E6550 2.33GHz processor, main memory of 2GB and Hard disk of 300GB. All programs are written using JAVA language (JSDK 1.6) in the operating system of windows XP. The parameters used in the experiments are listed in Table 2.

Table 2. Parameters Used in Experiments

Parameter	Setting
Number of total mobile devices	100
Cardinality of global relation	50K, 100K, ..., 300K
Cardinality of local relations	500, 1000 , ..., 3000
Dimensionality	2, 3, 4, 5
Ratio of skewed data	50%, 60%, 70% , ..., 90%

In our simulation, we set one hundred handsets. We vary the number of objects in a device from 500 to 3000. The default is 1000. Also, to measure the effect of the number of dimensions, we change the number of dimensions from two to five. The default dimension is five.

To show the performance according to the data skewness, we locate a subset of objects to the randomly selected region. The ratio of subset to whole data set changes from 50% to 90%. The default ratio is 70%.

We consider two performance measures. The first measure is the data reduction ratio (i.e., the efficiency of communication cost) among mobile devices. The second measure is the computation improvement (i.e. the average of the number of comparisons) since the filtering tuple reduces the number of comparisons. Our method is compared with [6] that motivates this work.

A. The costs of communication

In this experiment, we measure the reduction ratio of communication cost among mobile devices. The experimental results for communication cost are shown in Figure 8. *VDR* is the choosing criteria of the filtering tuple in [6]. *DPL* is our filtering tuple criteria.

Figure 8(a) shows the results of data reduction ratio with changing the number of objects. Data reduction ratio is ratio of filtered tuples by the filtering tuple in a local skyline set. *DPL* shows the larger data reduction ratio than *VDR*. Figure 8(b) shows the result of the data reduction ratio with changing the number of dimensions. Figure 8(c) shows the results of the data reduction ratio with changing the ratio of skewed data. Our approach outperforms the competitors in all experimental cases. Generally, as increases the number of objects or the number of dimensions, the reduction ratio reduces. But, as increases the skewed data ratio, the reduction ratio increases.

B. Effect of Local Computation

In this experiment, we show the execution time on single mobile device. The experimental results of execution time are shown in Figure 9. We focus on the average number of comparisons when skyline is computed since the comparisons take the largest time of the skyline query processing.

Figure 9(a) shows the results of the number of comparisons with changing the number of objects. Increment of *DPL* is smaller than that of *VDR*. Figure 9(b) shows the number of comparisons with changing the number of dimensions. Figure 9(c) shows the number of comparisons with changing the ratio of skewed data.

VDR has not the filtering step before computing skyline queries. In contrast, *DPL* has the filtering step before computing skyline. Thus, the number of comparisons of *DPL* is reduced significant compared to *VDR*.

VI. CONCLUSIONS

This paper addresses the skyline query processing in a distributed network environment. Each mobile device contains some portion of the data. Thus, a device communicates with other devices to retrieve whole data. To reduce the communication cost among mobile devices and the execution time on each single mobile device, we use the filtering tuple. With respect to the filtering tuple, the communication overhead and computation cost are changes. Thus, we propose a new selection strategy for the filtering tuple based on mode value. Before computing a skyline set, we prune out objects that dominated by the filtering tuple. So, we reduce the execution time by reducing the number of comparisons. Also, by eliminating objects that dominated by the filtering tuple, we reduce the communication cost.

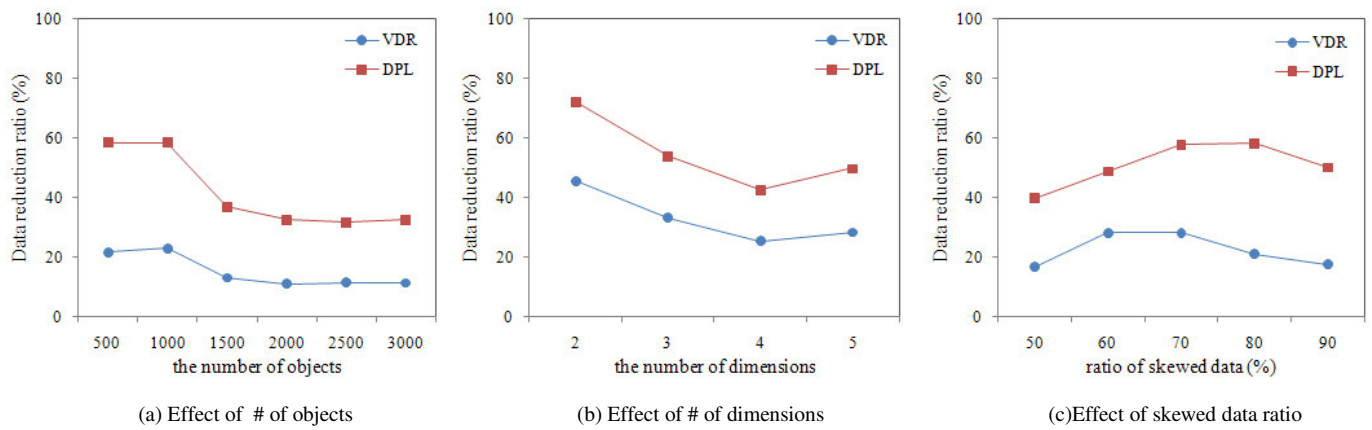


Figure 8. Data Reduction Rate on Local relation in static setting

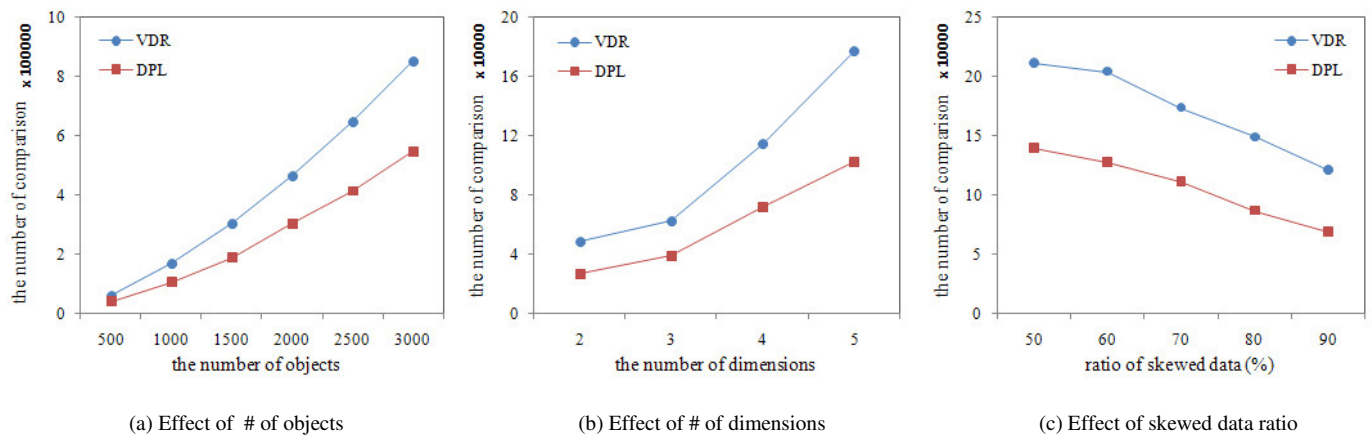


Figure 9. The number of comparison on Local relation in static setting

The effectiveness of our proposed method is validated by experiments with diverse environments. As for future research, we will investigate an efficient approximate algorithm for stream skyline query processing in the P2P environments.

ACKNOWLEDGMENT

This work was supported by national research foundation of Korea grant funded by the Korean government (grant number 2009-0072612)

REFERENCES

- [1] M. Gerla, C. Lindemann, A. Rowstron, "P2P MANET – New Research Issues," (2005) The Dagstuhl Research Online Publication Server [Online]. Available: <http://drops.dagstuhl.de/opus/volltexte/2005/213/>
- [2] Fusing ad hoc and P2P, *Pictures of the Future (Siemens Magazine for Research and Innovation)*, Spring, 2005.
- [3] S. Borzanyi, D. Kossmann, and K. Stocker. "The skyline operator," In Proc. IEEE ICDE, p. 421-430, 2001.
- [4] J. Chomicki, P. Godfery, J. Gryz, and D. Liang, "Skyline with presorting," In Proc. IEEE ICDE, p. 717-719, 2002.
- [5] W. T. Balke, U. Guntzer, and J. X. Zheng. "Efficient distributed skylining for web information systems," In EDBT, p. 256-273, 2004.
- [6] Z. Huang, C. S. Jensen, H. Lu, and B. C. Ooi, "Skyline Queries Against Mobile Lightweight Devices in MANETs," In Proc. IEEE ICDE, p. 66, 2006.
- [7] S. Wang, B. C. Ooi, A. K. H. Tung, and L. Xu, "Efficient skyline query processing on peer-to-peer networks," In Proc. IEEE ICDE, p.1126-1135, 2007.
- [8] D. Sun, S. Wu, J. Li, and A. K. H. Tung, "Skyline-Join Distributed Databases," IEEE ICDE, 2008.
- [9] B. Cui, H. Lu, Q. Xu, L. Chen, Y. Dai and Y. Zhou, "Parallel distributed processing of constrained skyline queries by filtering," IEEE ICDE, p. 546-555, 2008.
- [10] S. H. Yoon and C. Shahabi, "Distributed Spatial Skyline Query Processing in Wireless Sensor Networks," In IPSN, 2009.
- [11] S. Antony, P. Wu, D. Agrawal, and A. E. Abbadi, "Efficient Skyline Computation over ad-hoc Aggregations," IEEE ICDE, 2008.