The Top-k Skyline Query in Pervasive Computing Environments

Peng Pan, YuQing Sun,Qingzhong Li, ZhiYong Chen, Ji Bian
School of Computing Science and Technology
ShanDong University
JiNan, P.R.China
ppan, sun_yuqing, lqz,chenzy,bj@sdu.edu.cn

Abstract—**In pervasive computing environments, more and more applications or platforms based on mobile phone and PDA emerged. People could get what they want from these platforms anywhere. *Top-k* and *skyline* query are two methods to satisfy user's preferences. In many situations, we need to combine the two querying technologies to satisfy the user's preferences, which is called top-k skyline query. Based on a given pervasive computing scenario, we formulate a theoretical model for top-k skyline query, describe an algorithm that proceeds the skyline and top-k query synchronously. It shows better performance in experiment than algorithms proposed in [3].**

Keywords- Pervasive computing, skyline, top-k, top-k skyline

I. Introduction

Recently, pervasive computing is getting more and more researchers' attentions with the development and maturation of related technologies and theories gradually[1]. The ultimate goal of pervasive computing is to integrate the information spaces constructed by communication and computer and to integrate the physical spaces for people's lives and works [2].

With the development of mobile technology, more and more applications or platforms based on mobile phone and PDA emerged. People would get what they want from the information services provided by these platforms when submitting the query according to their preferences. To date, there are two major methods for query based on user's preferences: top-k and skyline. They are both used to solve the traditional problem of multi-objects optimization. Top-k query retrieves the best *k* objects that minimize a specific preference function, which transform the problem of the multi-objects optimization into single object one. For skyline, given a set of *d*-dimensional objects, the query returns the "best" objects that are not dominated by others. Differed from the top-k query, the skyline solves the original multi-objects with algorithms of multi-objects optimization. As the result is a set whose elements are not dominated mutually and the amount of the set may be large, it's unfeasible and worthless to present all the results to the users. We may refine the skyline results by a top-k query. An example is as follow:
Example 1:

A tourist is interested in the top-k restaurants with the best food , and these restaurants are the closest to certain spot and the cheapest.

The tourist can formulate the following top-k query:

```
        Select *
        From Guide
        Skyline of cheap(Price) max, close(Address, HotelAd)
max
        Order By max(quality(Food))
        Top k
```

Example 1 is a combination of skyline and top-k query. The function *cheap* is a score function, which indexes the restaurants on prices; *close* is an user-defined function which orders the distances between a certain spot and the restaurants; *quality* is an user-defined function which evaluates the food quality for each restaurant. In the process, the query executes the operator 'Skyline' to get the skyline results, and then executes the 'Order By…Top k' to get to the final results. [3] has proposed a algorithm for top-k skyline query. However, it is inefficient to separate the processes of skyline and top-k query into two isolated steps,. In this paper, we present a new algorithm combining the processes of skyline and top-k into an integrated process, and the experiments show better performance than the one in [3].

The paper is organized as follows. Section 2 describes a scenario in computing pervasive environment which motivates our research. Section 3 discusses related works. Section 4 defines the theoretical model for top-k skyline. Based on the scenario mentioned in section 2, section 5 describes an algorithm of top-k skyline query. Section 6 presents the experiments, and section 7 concludes.

II. Motivating Scenario

To illustrate the research focus and motivate the need for our discussion, consider the following scenario:

Sams is traveling to JiNan which is famous for the splendid springs in China. He wants to find a hotel with the best food, and these hotel are close to a famous spring – BaoTu spring and cheap. Using his PDA, he submits his preferences by visiting a mobile travel service platform. After executing the top-k skyline query operator as Example 1, the platform presents the user results. Figure 1 presents the architecture for the scenario.

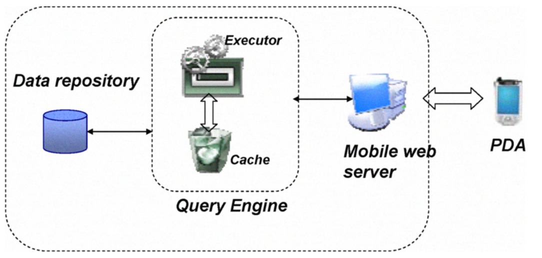

Figure 1. The Architecture of the mobile information service platform

In Figure 1, the Mobile web server provides a portal to the PDA, transfer the user's preference to top-k skyline query engine, and presents the results to user. Based on the data repository, the query engine executes the query operators including the top-k skyline algorithm depicted in this paper.

III. Related Works

In the literature of query answering algorithms proposed for *top-k* queries, the *Threshold Algorithm (TA)* is a representative[4] algorithm. It is generally applicable in database applications, but inefficient in large distributed networks. Therefore, several variations of TA have been proposed, such as the *Three-Phase Uniform-Threshold* algorithm (TPUT)[5], *BPA* and *BPA*2[6].

The term *skyline* has been proposed in the database literature [7] to refer to the secondary storage version of the maximal vector set problem. Since then, several algorithms have been proposed for skyline queries that include no preprocessing solutions (e.g., [8]), presorting solutions (e.g., [9]), and index-based solutions (e.g., [10], [11]). The partition-index algorithm presented in [11] motivated this paper.

Recently, solutions for the combination of *skyline* and *top-k* have been defined[12], [13], [14]. In general, these solutions calculate the first stratum or skyline with some sort of post-processing. However, without considering situations as in section 2, these solutions do not identify the *k* best answers. The authors of [15], [16] proposed the problem of selecting k skyline points so that the number of points, which are dominated by at least one of these k skyline points, is maximized. The most related works to this paper are investigated in [3], [17]. The authors attempt to solve the top-k skyline problem by proposing the models and algorithms.

IV. The model for Top-k Skyline query

Given a space *S* with *d* dimensions, $S = \{s_1, s_2 .... s_d\}$, a set *D* of points in *S*, $D = \{p_1, ...p_n\}$, i.e. each $p_i \in D$ is a point with *d* dimensions in *S*. We denote the value for the *i*th dimension of point *pi* by $p_i.s_j$.For each s_j, we assume there is a total order denoted by $\succ_j$ in its value of domain, where $\succ_j$ can be the relationship of '<' or '>' in terms of user's preference.

Definition 1: dominate

A point p_i is said to *dominate* another point p_j on *S* if and only if $\forall s_k \in S$, $p_i.s_k \succ p_j.s_k$, and $\exists s_t \in S$,$p_i.s_t \succ p_j.s_t$

Definition 2 : skyline,SP(D,S)

A point p_i is a skyline point in S, where p_i satisfies $\neg\exists p_j : p_j \neq p_i \wedge p_j$ dominating p_i

Definition 3: scoring function

The scoring function s on relation *R* is a function from *R* to real numbers, i.e. $R \mapsto \mathbb{R}$. The function s induces a preference relation $\succ_s$ and an indifference relation $\sim_s$ on *R*. For any two different tuples t_i and t_j

$t_i \succ_s t_j$ iff $s(t_i) > s(t_j)$

$t_i \sim_s t_j$ iff $s(t_i) = s(t_j)$

Definition 4: the set for top-k query on R

Given a relation *R*, a non-negative integer *k*, a scoring function s, a set for top-k query satisfying s on *R* is following tuples set *T*:

1 $T \subseteq R$

2 if |R| < k, $T = R$, otherwise $|T| = k$

3 $\forall t \in T, \forall t' \in R - T, t \succ_s t'$ or $t \sim_s t'$

Definition 5: the set for top-k query on skyline query

SP is a skyline set in terms of definition 2, *k* is a non-negative integer, and *f* is a scoring function on *SP*, a set for top-k query satisfying *f* on *SP* is a tuples set *SPT* as follows:

1 $SPT \subseteq SP$

2 if $|SP| < k$, $SPT = SP$, otherwise $|SPT| = k$

3 $\forall st \in SPT, \forall st' \in SP - SPT, st \succ_f st'$ or $st \sim_f st'$

Figure 2 is an example of the model whose dimensions of relation *R* are 2. The p_i is a skyline point, and q_i is the corresponding point in term of a scoring function *f*. The objective is to find the top *k* point satisfying the scoring *f* whose parameter is in the domain of skyline points.

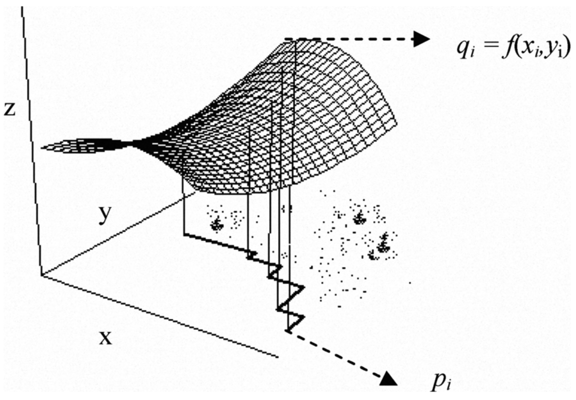

Figure 2. An example of model for top-k skyline query

V. Algorithm for Top-k Skyline query

A. An Example of Data Instances for Algorithm

In the data repository related to the user's query of the scenario in section 2, the original tuples are in the list of Table 1. To apply the skyline algorithm proposed in this paper, we need to normalize the values in each dimension. We set the ranks of each dimension value as given in Table 1. The dimension values are transformed by substituting the real value with its rank among all values. Table 2 is the result of the transformation.

Table 1. An Example of Data Instances

Name	short	Price (RMB)	Distance (m)
Guihe Crown Holiday	d	1350	1068
City of Spring	h	667	1169
YuQuan	p	580	1210
Qihui Weijing	k	1160	1354
Shunhe Tianxi	a	668	1380
Sofitel Silver Plaza	b	1725	1411
Good Friend	f	897	1949
GuiDu	c	620	1996
Yayue	o	238	1998
News Building	l	860	2034
ZhongHao	m	830	2068
Nanjiao	g	580	2212
Shandong Building	j	1280	2284
Rujia	i	199	2410
LongDu	e	1136	2818
ShunGeng villa	n	720	3101

Table 2. Result of Transformation

Name	short	Price (RMB)	Distance (m)	Price (rank)	Distance (rank)
Shunhe Tianxi	a	668	1380	7	5
Sofitel Silver Plaza	b	1725	1411	16	6
GuiDu	c	620	1996	5	8
Guihe Crown Holiday	d	1350	1068	15	1
LongDu	e	1136	2818	12	15
Good Friend	f	897	1949	11	7
Nanjiao	g	580	2212	4	12
City of Spring	h	667	1169	6	2
Rujia	i	199	2410	1	14
Shandong Building	j	1280	2284	14	13
Qihui Weijing	k	1160	1354	13	4
News Building	l	860	2034	10	10
ZhongHao	m	830	2068	9	11
ShunGeng villa	n	720	3101	8	16
Yayue	o	238	1998	2	9
YuQuan	p	580	1210	3	3

The algorithm partitions the 2-dimensional tuples into two lists in the way that tuple $t = (p_1,..p_d)$ is put into list i if the *ith* dimension value of tuple t is the minimum among all dimensions of t ,that is $p_i \le p_j \forall i \ge 1 \wedge i \ne j$. For example, tuple *c*("GuiDu") is put into list1, since its normalized dimension values are (5,8). Table 3 is the final lists. The tuples in the two lists can be indexed by any dimensional indexing structure. In this paper, we use B+-tree to index the tuples shown in Figure 3.

Table 3. Final Lists

List1		List2	
Point	Minimum distance	Point	Minimum distance
i	1	d	1
o	2	h	2
p	3	k	4
g	4	a	5
c	5	b	6
n	8	f	7
m	9	e	10
l	10		
j	14		

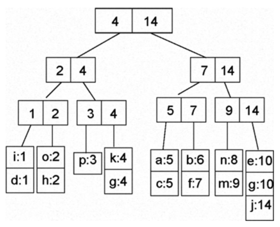

Figure 3. B+-tree for the Lists of Table 3

B. Algorithm for Top-k Skyline Query

The algorithm proposed in [3] divides the process into two unattached steps: skyline query and top-k query. On the results of the whole set of skyline, the algorithm run top-k query to get the final results, but it shows low efficiency. The algorithm in this paper may get the final top-k skyline results without having to scan all the data spaces, and by normalizing the values in each dimension, it processes the skyline query and top-k query synchronous. As soon as the k top-k skyline tuples emerge, the query process would be ended, and it's unnecessary to travel the whole skylines set. Figure 4 is the detailed algorithm where .

fi is a flag to determine whether the *i*th dimension still needs to be searched. If fi is set to false, it means that all subsequent tuples are dominated by some tuple, and the i*th* dimension needs not to be searched any more,

maxValue(t) returns the maximum value among all dimensions of tuple *t*, which satisfies $t.s_i \ge t.s_j (j \in [1,d] \wedge j \ne i)$,

minValue(t) returns the minimum value among all dimensions of tuple *t*, which satisfies $t.s_i \le t.s_j (j \in [1,d] \wedge j \ne i)$,

traverseTreeForMin(root,i) traverses the B+-tree to obtain the tuple with the minimum value for the *i*th dimension,

getNextNode(t) returns the next node for some node in a B+-tree,

computePartitionSkyline(P) is used to compute the skyline sets of the nodes set P, which may adopt any algorithm of skyline at present,

ComputeNewSkyline(Sj,S) computes the new skyline from S_j and S,

computerTop-k gets the top-k results from existing skyline set.

```
Input : Dataset D
Input : B+-tree B
Output : Skyline S
for i = 1 to d
    $f_i \leftarrow$ True
    $t_i \leftarrow$ traverseTreeForMin(root,i)
    max[i] $\leftarrow$ maxValue($t_i$)
    min[i] $\leftarrow$ minValue($t_i$)
$mn \leftarrow \min_{i=1}^{d} \max[i]$
$mx \leftarrow \min_{i=1}^{d} \min[i]$
for i =1 to d
    if mn < max[i]
        $f_i \leftarrow$ False
j $\leftarrow$ 1
$S \leftarrow \varnothing$
$T \leftarrow \varnothing$
while there are some partitions to be searched
    for i=1 to d
        if min[i] == mx
            $P_j \leftarrow t_i$
            $S \leftarrow \varnothing$
            $t_i \leftarrow$ getNextNode($t_i$)
            while (minValue($t_i$) == mx)
                mn $\leftarrow$ min(mn, maxValue($t_i$))
                $P_j \leftarrow P_j \cup t_i$
                $t_i \leftarrow$ getNextNode($t_i$)
            min[i] $\leftarrow$ minValue($t_i$)
    Sj $\leftarrow$ computePartitionSkyline($P_j$)
    $S \leftarrow S \cup computeNewSkyline(S_j, S)$
    T $\leftarrow$ *computerTop-k*(S)
    if |T| == k
        end
    j $\leftarrow$ j + 1
    mx $\leftarrow \min_{i=1}^{d} \min[i]$
    for i=1 to d
        if mn < min[i]
            $f_i \leftarrow$ False
```

Figure 4. the Algorithm for Top-k Skyline Query

VI. EXPERIMENT

We now present experimental results to illustrate the discussion in Section 5. Experiments were performed on a 2GHz Pentium IV Windows desktop with 1 GB of RAM and 60 GB of hard disk space. The algorithm in Section 5 was implemented in C, and it accesses database through cursor operations.

The objective of the experiment is to compare the executing efficiency of the top-k skyline algorithm in this paper with the one proposed in [3] and to reveal the effect of different numbers of dimensions on execution time.

We take 10000 tuples as tested tuple vectors, and run two algorithms respectively by using different parameter values of dimensions from 1 to 10. Figure 5 is the results based on the experiment. The curve linked by triangle points which is represented by B-algorithm presents the performance of the algorithm in [3], the curve with square points labeled by N-aglorithm shows the performance of this paper's algorithm.

Since increasing the number of dimensions would produce more skyline tuples than lower dimension, more times would cost on proceeding the skyline tuples. Moreover, when the number of dimensions is up to certain point, the efficiency of both two algorithms will decrease rapidly. However, the algorithm proposed in this paper need not to scan all the tuples to get the final top-k result, which shows better performance than the algorithm proposed in [3] does.

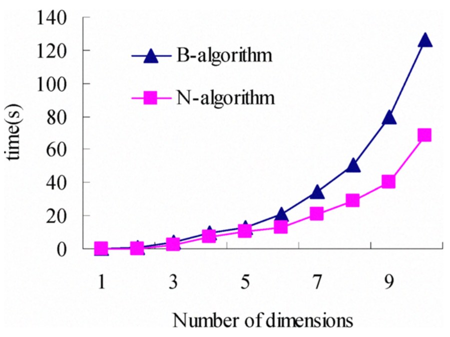

Figure 5. the Experiment for B-algorithm and N-aglorithm

VII. Conclusion and Future Works

This paper has discussed the problems related to the combination of top-k query and skyline query including theoretical model and algorithm. Based on a given scenario in pervasive computing environment, we have formulated a theoretical model for top-k skyline query, demonstrated the algorithm. Finally, we evaluated the algorithm which shows better performance.

Many other problems need to be solved and furthered. We need to find better way to improve the algorithm's performance.

References

[1] Mark Weiser, "The Computer for the 21st Century", Scientific American. Sep. 1991, 265(3): 94 – 104.

[2] M.Satyanarayanan, "Pervasive Computing: Vision and Challenges", IEEE Personal Communications, Aug. 2001,vol. 8, no. 4, pp. 10-17.

[3] Brando C, Goncalves M, Gonzalez V, "Evaluating top-k skyline queries over relational databases", In Proceedings of International Conference on Database and Expert Systems Applications. San Jose, USA: Springer, 2007. 254-263

[4] R.Fagin, A.Lotem, M.Naor, "Optimal aggregation algorithms for middleware Source", in Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, 2001, pp. 102-113.

[5] Pei Cao, Zhe Wang, "Efficient Top-K query calculation in distributed networks", in Proceedings of the 23rd Annual ACM Symposium on Principles of Distributed Computing, 2004, pp. 206-215.

[6] R.Akbarinia, E.Pacitti, P.Valduriez , "Best position algorithms for top-k queries", in Proceedings of the 33rd international conference on VLDB2007,pp. 495-506.

[7] S. Bdrzsdnyi, D. Kossmann, and K. Stocker, "The Skyline Operator," in Proceedings of the International Conference on Data Engineering(ICDE), 2001,pp.235-254

[8] P. Godfrey, R. Shipley, and J. Gryz, "Maximal Vector Computation in Large Data Sets," in Proceedings of the International Conference on Very Large Data Bases, VLDB, 2005.

[9] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, "Skyline with Presorting," in Proceedings of the International Conference on Data Engineering(ICDE), 2003, pp. 717- 719

[10] D. Kossmann, F. Ramsak, and S. Rost, "Shooting Stars in the Sky: An Online Algorithm for Skyline Queries," in Proceedings of the International Conference on Very Large Data Bases, VLDB, 2002,pp. 275 - 286

[11] K.-L. Tan, P.-K. Eng, and B. C. Ooi, "Efficient Progressive Skyline Computation," in Proceedings of the International Conference on Very Large Data Bases, VLDB, 2001,pp. 301 - 310 .

[12] Balke, W-T., G¨untzer, U. "Multi-Objective Query Processing for Database Systems", In Proceedings of the International Conference on Very Large Databases (VLDB), pp. 936–947

[13] Goncalves, M., Vidal, M.E, "Preferred Skyline: A Hybrid Approach Between SQLf and Skyline", In Andersen, K.V., Debenham, J., Wagner, R. (eds.) DEXA 2005. LNCS, vol. 3588, pp. 375–384. Springer, Heidelberg (2005)

[14] Lo, E., Yip, K., Lin, K-I., Cheung, D, "Progressive Skylining over Web-Accessible Databases", Journal of Data and Knowledge Engineering 57(2), pp.122–147 (2006)

[15] X. Lin, Y. Yuan, Q. Zhang, Y Zhang. "Selecting Stars: The k MostRepresentative Skyline Operator", In Proc. of the Int. IEEE Conf. on Data Engineering (ICDE), Istanbul, Turkey, 2007,pp. 86-95

[16] Man Lung Yiu, Nikos Mamoulis, "Efficient Processing of Top-k Dominating Queries on Multi-Dimensional Data", vldb07,pp.483-494

[17] Goncalves, M., Vidal, M.E., "Top-k Skyline: A Unified Approach", In Proceedings of OTM (On the Move) 2005 PhD Symposium, pp. 790–799 (2005)J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.