Skyline Recommendation in Distributed Networks

Zhenhua Huang[1], Jiawen Zhang[1], Zheng Liu[1], Bo Zhang[2], and Dong Wang[3]
[1]School of Electronics and Information, Tongji University, China
[2]College of Information Mechanical and Electrical Engineering, Shanghai Normal University, China
[3]College of Computer Science and Information Engineering, Shanghai Institute of Technology, China

Abstract: *Skyline recommendation technology has recently received a lot of attention in the database community. However, the existing works mostly focus on how to obtain skyline objects from fine-grained data in centralized environments. And the time cost of skyline recommendation will increase exponentially as the number of data and skyline recommendation instructions increases, which will seriously influence the recommendation efficiency. Motivated by the above fact, this paper proposes an efficient algorithm Skyline Recommendation Algorithm in Distributed Networks (SRADN) in Super-Peer Architecture (SPA) distributed networks to handle multiple subspace skyline recommendations by prestoring the set of skyline snapshots under the cost constraint of maintenance and communication. The proposed SRADN algorithm fully considers the characteristic of storage and communication of SPA networks, and uses the map/reduce distributed computation model. The SRADN algorithm can quickly produce the optimal set of skyline snapshots through the following two phases: Heuristically constructing the initial set of snapshots, and adjusting the set of snapshots based on the genetic algorithm. The detailed theoretical analyses and extensive experiments demonstrate that the proposed SRADN algorithm is both efficient and effective.*

Keywords: *Skyline recommendation, distributed networks, map/reduce, genetic algorithm.*

Received August 30, 2014; accepted December 16, 2014

1. Introduction

Skyline recommendation technolgy has attracted much attention recently since it is used in many applications, such as big data analysis, high-dimensional data visualization, and multi-criteria decision making [21]. Given a set of objects $\mathcal{R}$= $\{p_1, \ldots, p_n\}$, each object p_i ($i \in [0, n]$) has m dimensions $F=\{d_1, \ldots, d_m\}$, the skyline recommendation over subspace $U \subseteq F$ is to return the objects that are not dominated by any other objects restricted to U. In fact, the preference function "dominate" can be defined in any way as long as it is monotone on U [14]. It is not difficult to see that for a set of objects including m dimensions, it has at most 2^m-1 different subpace skyline recommendations [10].

Recently, various techniques have been proposed for processing the skyline recommendation. Literature [2] first presented the concept of skyline recommendation, and proposed two feasible recommendation algorithms: Block Nested Loop (BNL) and Divide and Conquer (DC). BNL essentially compares each object in the database with all the other objects, and outputs the one only if it is not dominated in any case. DC divides the dataset into several partitions that could fit in memory. The skyline objects in all partitions are computed separately using a main-memory algorithm, and then merged to produce the final result. Based on BNL, Literature [5] designed the Sort First Skyline (SFS) algorithm which sorts the input data according to a preference function, after which the skyline object could be found in another pass over the sorted list. Literature [8] theoretically gived the time cost of the algorithms BNL, DC and SFS under the assumption of independent distribution, and proposed an External-Sort Algorithm (ESA) to improve the recommendation efficiency. Specially, ESA could reduce the time complexity of skyline recommendation to $O(|\mathcal{R}|\log|\mathcal{R}|+m|\mathcal{R}|)$. Literature [11] integrated k-means clustering into skyline recommendation, and returns k "representative" and "diverse" skyline objects to users. Literature [15] was based on the model of possible world instances [16] and proposed two efficient algorithms Bottom-Up Agorithm (BUA) and Top-Down Agorithm (TDA) to process the skyline recommendation on uncertain data. BUA utilized the R-tree index structure [12], and returned all skyline objects whose probabilities are greater than the threshold ξ by three phases: bounding, pruning and refining. While TDA organized all uncertain data objects as a partition tree [22], and used three effective properties of partition tree to decrease the comparison number between objects.

As the wide use of distributed networks, Literature [19] first considered processing skyline recommendation in Super-Peer Architecture (SPA) [23] distributed networks, and proposed the concept of extended skyline set to reduce the cost of data transmission. Based on literature [19], literature [6] presented the Multidimensional Routing Indices (MRI) index structure to decrease the number of network nodes which take part in the skyline recommendation. The MRI index structure could further reduce the cost of data transmission. Literature [13] identified the drawbacks of the methods in literatures [19], and

proposed an efficient algorithm to improve the skyline recommendation performance in SPA distributed networks. The algorithm used bloom filter [7] to reduce the cost of data transmission, and utilized the regular grid index [17] to organize the data objects and thus could remarkably improve the computation efficiency for skyline recommendation. Literature [18] introduced the concept of Vertical Partition Skyline (VPS) in SPA distributed networks. VPS was an algorithmic framework that includes two phases. In the first phase, VPS searched for an anchor point p_{anc} which dominates, and hence eliminates, a large number of objects. And in the second phase, starting with p_{anc}, VPS constructed incrementally a pruning area using an interesting union-intersection property of dominance regions. The network nodes did not transmit those objects falling within the pruning area in their local subspace, which could evidently improve the skyline recommendation performance.

To the best of our knowledge, the existing skyline recommendation algorithms in SPA distributed networks use fine-grain basic data as the input parameter. Hence, as the data volume and dimensionality increase, network communication cost and CPU cost will exponentially increase. Accordingly, it will seriously influence the skyline recommendation efficiency. Due to the appearance of high-capacity cheap disks, we can prestore w skyline snapshots $SN=\{s_1,\ldots, s_w\}$ in SPA distributed networks to efficiently process u subspace skyline recommendations $SR=\{sr_1,\ldots, sr_u\}$. On the other hand, when basic data is changed, the w prestoring skyline snapshots need to be periodically updated, which will need extra maintenance cost for these skyline snapshots. And transferring the skyline snapshots from storage nodes to computation nodes needs extra network communication cost.

Based on the above facts, in this paper, we propose Skyline Recommendation Algorithm in Distributed Networks (SRADN), an efficient algorithm in SPA distributed networks to process u subspace skyline recommendations $SR= \{sr_1, \ldots, sr_u\}$ by prestoring w skyline snapshots $SN= \{s_1, \ldots, s_w\}$ under the cost constraint of maintenance and communication. Our SRADN algorithm utilizes the map/reduce distributed computation model [1] and can quickly produce the optimal set of skyline snapshots through the following two phases: heuristically constructing the initial set of snapshots, and adjusting the set of snapshots based on the genetic algorithm. The detailed theoretical analyses and extensive experiments demonstrate that our SRADN algorithm is both efficient and effective.

The rest of the paper is organized as follows: Section 2 gives the problem description of our work. Section 3 presents the approach for exact selection of optimal prestoring skyline snapshots. In section 4, we propose the SRADN algorithm to fast produce the optimal set of skyline snapshots using the map/reduce distributed computation model. We present the experimental study in section 5. Finally, section 6 concludes the paper with directions for future work.

2. Problem Description

Without loss of generality, we let the SPA distributed network $\mathfrak{I}$ include λ storage nodes $N_g^{(1)}, \ldots, N_g^{(\lambda)}$, and the computation node in $\mathfrak{I}$ be N_c. Assume that u subspace skyline recommendations $SR=\{sr_1,\ldots, sr_u\}$ are submitted on N_c, and candidate skyline snapshots $CSN= \{s_1, \ldots, s_\gamma\}$ are distributedly stored on $N_g^{(1)}, \ldots, N_g^{(\lambda)}$.

First of all, we give three cost models for skyline recommendation in SPA distributed networks.

1. *Computation Cost Model*: the time cost of obtaining the result of subspace skyline recommendation sr from skyline snapshot s, is denoted as t_{sr}^{s} .It includes two parts: the I/O cost of transferring s from the disk to memory, denoted as t_s; and the CPU cost of obtaining the result of sr from s, denoted as $t_{s\to sr}$.

The I/O cost t_s can be easily obtained and is expressed below Equation 1:

$$t_s = \frac{size(s)}{block\_size} \cdot t_{I/O}^{block} \tag{1}$$

Where $size(s)$ is the size of s, $block\_size$ is the block size, and $t_{I/O}^{block}$ is the time cost of transferring a block.

We then give the CPU cost $t_{s\to sr}$ as follows [3]. Without loss of generality, we assume that the subpace of sr is V, and let $v=|V|$.

- *Theorem 1*. Assume that s satisfies the joint distribution function $F(\bar{x})$ and the joint density function $f(\bar{x})$ on V, where $\bar{x} = (x_1, \ldots, x_v)$. Then the expected value $E(s, v)$ of objects returned by sr can be denoted as Equation 2:

$$|s| \times \int_{[0,1]^v} f(\bar{x})(1-F(\bar{x}))^{|s|-1} d\bar{x} \tag{2}$$

- *Theorem 2*. Assume that s satisfies the joint distribution function $F(\bar{x})$ and the joint density function $f(\bar{x})$ on V, where $\bar{x} = (x_1, \ldots, x_v)$. Then the CPU cost $t_{s\to sr}$ can be denoted as Equation 3:

$$\sum_{x=2}^{|s|} E(x-1, v) \times E(x-1, v+1) \Big/ x-1 \tag{3}$$

2. *Maintenance Cost Model*: the time cost of updating s when the basic data $\mathfrak{R}$ changed. We let the subspace of s be V, and assume s is stored on the storage node $N_g^{(i)}$ $(1\le i\le \lambda)$ which also stores γ skyline snapshots $s^{(1)}, \ldots, s^{(\gamma)}$. According to the literature [9], we can know that only those skyline snapshots whose subspaces include V can be used to update s. We select s_min, the skyline snapshot which needs the minimal time cost to update s. It is

not difficult to see that the time cost of using *s_min* to update *s* equals the one of obtaining the result of *s* from *s_min*, i.e., $t_s^{s\_min}$ (see the Equations 1, 2, 3).

3. *Communication Cost Model*: the time cost of transferring *s* from the storage node $N_g^{(i)}$ ($1 \le i \le \lambda$) to the computation node N_c, is denoted as nt_s. The communication cost can be expressed below Equation 4:

$$nt_s = \frac{size(s)}{TS_{N_g^{(i)} \to N_c}} \tag{4}$$

Where $TS_{N_g^{(i)} \to N_c}$ is the network bandwidth between $N_g^{(i)}$ and N_c.

Based on the computation cost model, we can select and prestore *w* ($w<\gamma$) skyline snapshots $SN=\{s_1, \ldots, s_w\}$ from candidate ones, and the time cost of processing *u* subspace skyline recommendations $SR=\{sr_1,\ldots, sr_u\}$ can be expressed as Equation 5:

$$comCost(SR)=\sum_{i=1}^{u} t_{sr_i\_\min \to sr_i} \tag{5}$$

Where sr_i_min belongs to *SN* and needs the minimal time cost to obtain the result of sr_i.

On the other hand, based on the maintenance cost and the communication cost, the time cost of maintaining and transferring *SN* can be expressed as Equation 6:

$$mtCost(SN)=\sum_{i=1}^{w} (t_{s_i}^{s_i-\min} + nt_{s_i}) \tag{6}$$

- *Problem Definition*: In this paper, given *u* subspace skyline recommendations $SR=\{sr_1,\ldots, sr_u\}$ and the user threshold of maintenance and communication cost *userCost*, our goal is to select and prestore the optimal *w* ($w<\gamma$) skyline snapshots $SN=\{s_1,\ldots, s_w\}$ from γ candidate ones such that $mtCost(SN) \le userCost$ and $comCost(SR)$ is minimal.

3. Exact Selection of Optimal Prestoring Skyline Snapshots

Given the user threshold *userCost*, in order to exactly select the optimal *w* skyline snapshots, we need traverse the exponential combinations of skyline snapshots. And hence it is an NP-hard problem, which can be proved in Theorem 3.

- *Theorem 3*. Assume there exists *u* subspace skyline recommendations $SR=\{sr_1,\ldots, sr_u\}$ and γ candidate skyline snapshots $CSN=\{s_1,\ldots, s_\gamma\}$ in the SPA distributed network. Given the user threshold of maintenance and communication cost *userCost*, it is an NP-hard problem to select *w* ($w<\gamma$) skyline snapshots $SN=\{s_1,\ldots, s_w\}$ from *CSN* such that $comCost(SR)$ is minimal.
- *Proof*. The time cost of obtaining *SN* is mainly determined by the search process of combinations of skyline snapshots. It is not difficult to see that each combination of skyline snapshots needs the capability to handle all *u* subsapce skyline recommendations. In the following part, we determine the time complexity of exactly obtaining *SN* by analyzing the number of combinations of skyline snapshots.
- For *w*=1, the number of combinations of skyline snapshots Equation 7:

$$INS^{(1)}=C_\gamma^1 \cdot C_u^u = \gamma \tag{7}$$

- For *w*=2, the number of combinations of skyline snapshots (Equation 8):

$$INS^{(2)}=C_\gamma^2 \cdot (C_u^1 + \ldots + C_u^i + \ldots + C_u^{u-1}) = C_\gamma^2 \cdot (2^u - 2) \tag{8}$$

- For *w*=*t*, the number of combinations of skyline snapshots (Equation 9):

$$\begin{aligned} INS^{(t)}=C_\gamma^t \cdot \{&(C_u^1 \cdot C_{u-1}^1 \ldots C_{u-t+1}^1 + \ldots + C_u^1 \cdot C_{u-1}^1 \ldots C_{u-t+1}^{n_2-t} + \ldots \\ &+ C_u^1 \cdot C_{u-1}^{u-t+2} \cdot C_{t-2}^1 \cdot C_{t-3}^1 \ldots \cdot C_1^1) + \ldots + (C_u^i \cdot C_{u-i}^1 \ldots C_{u-t+1}^1 \\ &+ \ldots + C_u^i \cdot C_{u-i}^1 \ldots C_{u-t+1}^{u-t} + C_u^i \cdot C_{u-i}^{u-i-t+2} \cdot C_{t-2}^1 \cdot C_{t-3}^1 \ldots \cdot C_1^1) \\ &+ \ldots + C_u^{u-t+1} \cdot C_{t-1}^1 \cdot C_{t-2}^1 \cdot C_{t-3}^1 \ldots \cdot C_1^1\} = C_\gamma^t \cdot (t^u - t) \\ &\ldots \end{aligned} \tag{9}$$

So, the time complexity $O(\gamma, u)$ of exactly obtaining *SN* is $INS^{(1)}+\ldots+INS^{(\gamma)} = \sum_{i=1}^{\gamma} C_\gamma^i \cdot (i^u - i)$. From $O(\gamma, u)$, we can know that exactly obtaining *SN* needs exponential time complexity, which belongs to NP problem. On the other hand, for a given combination of skyline snapshots *IRS* including ϖ skyline snapshots $\{s_1, \ldots, s_\varpi\} \subseteq CSN$, deciding whether if *IRS* is optimal can be reduce to the minimum cover problem of weighted directed bipartite graph $G(IRS, SR, W)$ [20], where *W* is the computation cost from *IRS* to *SR*. According to the graph theory [4], we can know that the minimum cover problem of weighted directed bipartite graph is an NP-hard problem. And hence exactly obtaining *SN* is an NP-hard problem.

From Theorem 3, we can see that exactly obtaining optimal *w* skyline snapshots needs massive CPU time cost. Hence, in the next section, we propose an efficient algorithm SRADN to fast achieve the approximate optimal solution.

4. The SRADN Algorithm

The core idea of SRADN is to use the map/reduce distributed computation model and fast produce the approximate optimal set of skyline snapshots through two phases: Heuristically constructing the initial set of snapshots, and adjusting the set of snapshots based on the genetic algorithm.

The implementation process of SRADN can be shown in Algorithm 1.

Algorithm 1: SRADN.

Input: candidate skyline snapshots CSN={s_1,…, s_γ}, subspace skyline recommendations SR={sr_1,…, sr_u}, the user threshold userCost;
Output: the approximate optimal solution ASN.
Begin
1. Construct CSN's corresponding key-value set $KY^{(CSN)}$={<'s'+i, s_i>|i∈[1,γ]};
2. Construct SR's corresponding key-value set $KY^{(SR)}$={<'sr'+j, sr_j>|j∈[1,u]};
3. Divide $KY^{(CSN)}$ into m parts $KY^{(CSN)}{}_1$,…, $KY^{(CSN)}{}_m$;
/ m is the user parameter */*
4. Divide $KY^{(SR)}$ into m parts $KY^{(SR)}{}_1$,…, $KY^{(SR)}{}_m$;
5. For λ=1 to m Do
6. SI_λ←$KY^{(CSN)}{}_\lambda$∪$KY^{(SR)}{}_\lambda$;
7. {<s, sr>|s∈$KY^{(CSN)}{}_\lambda$∧sr∈$KY^{(SR)}{}_\lambda$}←map(SI_λ);
/ s∈$KY^{(CSN)}{}_\lambda$, sr∈$KY^{(SR)}{}_\lambda$ and can be processed by s */*
8. Let the partition function $\overline{f}$ equal (i mod n);
/ n is the number of computers used to execute the reduce function */*
9. ASN←∅;
10. For x=1 to n Do / parallel processing */*
11. {<s', SR'>}←reduce({<s, sr>});
/ s'∈$KY^{(CSN)}{}_x$, SR'⊆$KY^{(SR)}{}_x$ and includes all subspace skyline recommendations processed by s' */*
12. ASN←ASN∪{s'|SR'≠∅};
13. Return ASN.
End

In Algorithm 1, SRADN first constructs two key-value sets $KY^{(CSN)}$ and $KY^{(SR)}$. In $KY^{(CSN)}$, each pair of key-value consists of a skyline snapshot ID and its corresponding entity; while in $KY^{(SR)}$, each pair of key-value consists of a subspace skyline recommendation ID and its corresponding entity (Lines 1 and 2). Then based on the user parameter *m*, SRADN divides $KY^{(CSN)}$ into *m* parts $KY^{(CSN)}{}_1$,…, $KY^{(CSN)}{}_m$, and also divides $KY^{(SR)}$ into *m* parts $KY^{(SR)}{}_1$,…, $KY^{(SR)}{}_m$ (Lines 3 and 4). The map function (Line 7) takes $KY^{(CSN)}{}_\lambda$∪$KY^{(SR)}{}_\lambda$ as the input parameter, and returns the intermediary key-value set {<*s*, *sr*>} through two-phase optimization process, where *s* is the skyline snapshot in $KY^{(CSN)}{}_\lambda$ and *sr* is the subspace skyline recommendation in $KY^{(SR)}{}_\lambda$ whose result can be obtained from *s*. The reduce function (Line 11) classifies the intermediary key-value set {<*s*, *sr*>}, and for each skyline snapshot *s*, outputs all subspace skyline recommendations whose results can be obtained from *s*. Finally, SRADN filters those skyline snapshots which are not used to handle any subspace skyline recommendations, and returns the remaining ones to users (Lines 12 and 13).

The map and reduce functions can be shown in Algorihms 2 and 3.

Algorithm 2: The map function.

Input: the key-value set $KY^{(CSN)}$={<skyline snapshot ID, skyline snapshot entity>}, the key-value set $KY^{(SR)}$= {<subspace skyline recommendation ID, subspace skyline recommendation entity>};
Output: the intermediary key-value set $KY^{(int)}$.
Begin
1. $KY^{(int)}$←∅;
2. mapCost←userCost/m;
3. rootS←the root skyline snapshot which can process all subspace skyline recommendations in $KY^{(SR)}$;
4. If mtCost({rootS})>mapCost Then Return NULL;
/ mtCost({rootS}) is the time cost of maintaining and transferring {rootS}, see Equation (6) */*
5. Else
6. For ∀<sr_id, sr_ent>∈$KY^{(SR)}$ Do
7. $KY^{(int)}$←$KY^{(int)}$∪{<rootS, sr_ent>};
8. $KY^{(int)}$_I←SRADN_I($KY^{(int)}$, $KY^{(CSN)}$, $KY^{(SR)}$);
/ Phase 1: heuristically constructing the initial set of snapshots */*
9. $KY^{(int)}$_II←SRADN_II($KY^{(int)}$_I);
/ Phase 2: adjusting the set of snapshots based on the genetic algorithm */*
10. $KY^{(int)}$←$KY^{(int)}$_II;
11. Return $KY^{(int)}$.
End

Algorithm 3: the reduce function.

Input: the intermediary key-value set $KY^{(int)}$={<skyline snapshot entity, subspace skyline recommendation entity>};
Output: the key-value set KY.
Begin
1. SN←the set of skyline snapshot entities in $KY^{(int)}$;
2. For ∀s∈SN Do
3. $sr^{(s)}$←∅;
4. For ∀<s, sr>∈$KY^{(int)}$ Do
5. $sr^{(s)}$←$sr^{(s)}$∪{sr};
6. KY←∅;
7. For ∀s∈SN Do
8. KY←KY∪{<s, $sr^{(s)}$>};
9. Return KY.
End

In Algorithm 2, the map function has two tasks, i.e., two optimization phases: Heuristically constructing the initial set of snapshots (Line 8), and adjusting the set of snapshots based on the genetic algorithm (Line 9). And two optimization phases SRADN_I and SRADN_II can be implemented as Algorithms 4 and 5.

Algorithm 4: SRADN_I.

Input: the key-value set $KY^{(int)}$={<skyline snapshot entity, subspace skyline recommendation entity>}, the key-value set $KY^{(CSN)}$={<skyline snapshot ID, skyline snapshot entity>}, the key-value set $KY^{(SR)}$={<subspace skyline recommendation ID, subspace skyline recommendation entity>};
Output: the key-value set $KY^{(int)}$_I.
Begin
1. TempS←{rootS};
2. SN←the set of skyline snapshot entities in $KY^{(CSN)}$;
3. For ∀<sr_id, sr_ent>∈$KY^{(SR)}$ Do
4. $s \leftarrow \min_{s\in SN} t^s_{sr\_ent}$;
/ $t^s_{sr\_ent}$ is the time cost of obtaining the result of sr_ent from s, see Equations (1)-(3) */*
5. If mtCost(TempS∪{s})≤mapCost Then
6. $KY^{(int)}$←$KY^{(int)}$∪{<s, sr_ent>}
-{<rootS, sr_ent>};
7. TempS←TempS∪{s};

8. $KY^{(int)}\_I \leftarrow KY^{(int)}$;
9. Return $KY^{(int)}\_I$.
End

Algorithm 5: SRADN_II.

Input: the key-value set $KY^{(int)}\_I$={<skyline snapshot entity, subspace skyline recommendation entity>}, the key-value set $KY^{(CSN)}$={<skyline snapshot ID, skyline snapshot entity>}, the key-value set $KY^{(SR)}$= {<subspace skyline recommendation ID, subspace skyline recommendation entity>};
Output: the key-value set $KY^{(int)}\_II$.
Begin
1. $SN \leftarrow$the set of skyline snapshot entities in $KY^{(CSN)}$;
2. $\zeta \leftarrow$the number of subspace skyline recommendation entities in $KY^{(SR)}$;
3. $VS \leftarrow \varnothing$;
4. $f(VS) \leftarrow 0$; / initialize the fitness function */*
5. For $\forall s \in SN$ Do
6. Construct the corresponding bit vector $V^{(s)}$ of s whose length equals ζ;
7. For x=1 to ζ Do
8. $sr_x \leftarrow$the x-th subspace skyline recommendation entity in $KY^{(SR)}$;
9. If $<s, sr_x> \in KY^{(int)}\_I$ Then
10. $V^{(s)}[x]=1$; $f(VS) \leftarrow f(VS) + t^s_{sr_x}$;
11. Else $V^{(s)}[x]=0$;
12. $VS \leftarrow VS \cup \{V^{(s)}\}$;
/ crossover : Lines 13-23 */*
13. For i=1 to $\lfloor |VS|/2 \rfloor$ Do
14. $\overline{VS} \leftarrow VS$;
15. $V^{(s)} \leftarrow$the corresponding bit vector of i-th skyline snapshot s;
16. $V^{(s')} \leftarrow$the corresponding bit vector of (i+1)-th skyline snapshot s';
17. Randomly select two exchange points a, b of $V^{(s)}$ and $V^{(s')}$;
18. Visit SN and obtain the first pairs (sn, sn') of skyline snapshots which satisfys: $V^{(sn)}[a, b] \vee V^{(sn')}[a, b] = V^{(s)}[a, b] \vee V^{(s')}[a, b]$; / $V^{(sn)}[a, b]$ is the bit vector between a and b of $V^{(sn)}$ */*
19. Exchange between $V^{(s)}[a, b]$ and $V^{(sn)}[a, b]$;
20. Exchange between $V^{(s')}[a, b]$ and $V^{(sn')}[a, b]$;
21. $SN' \leftarrow \{s | s \in SN \wedge \exists i, V^{(s)}[i]=1\}$;
22. If $mtCost(SN') \leq mapCost$ and $f(\overline{VS}) < f(VS)$ Then
23. $VS \leftarrow \overline{VS}$;
/ mutation: Lines 24-39 */*
24. For $i=\lfloor |VS|/2 \rfloor + 1$ to $|VS|$ Do
25. $V^{(s)} \leftarrow$the corresponding bit vector of i-th skyline snapshot s;
26. Randomly select two exchange points a, b of $V^{(s)}$;
27. For j=a to b Do
28. If $V^{(s)}[j]=0$ Then
29. $V^{(s)}[j]=1$;
30. Visit $\overline{VS}$ and obtain the first vector $V^{(sn)}$ whose j-th bit equals 1;
31. $V^{(sn)}[j]=0$;
32. Else
33. $V^{(n)}[j]=0$;
34. Visit SN and obtain the skyline snapshot sn which can process j-th subspace recommendation;
35. $V^{(sn)} \leftarrow$ the corresponding bit vector of sn;
36. $V^{(sn)}[j]=1$;
37. $SN' \leftarrow \{s | s \in SN \wedge \exists i, V^{(s)}[i]=1\}$;
38. If $mtCost(SN') \leq mapCost$ and $f(\overline{VS}) < f(VS)$ Then
39. $VS \leftarrow \overline{VS}$;
40. $KY^{(int)}\_II \leftarrow \varnothing$;
41. For $\forall V^{(s)} \in VS$, $\forall x \in V^{(s)}$ Do
42. If $V^{(sn)}[x]=1$ Then
43. $sr_x \leftarrow$the x-th subspace skyline recommendation entity in $KY^{(SR)}$;
44. $KY^{(int)}\_II \leftarrow KY^{(int)}\_II \cup \{<s, sr_x>\}$;
45. Return $KY^{(int)}\_II$.
End

In Algorithm 4, for each subspace skyline recommendation entity *sr_ent*, SRADN_I first computes the time cost of obtaining the result of *sr_ent* from every skyline snapshot, and chooses the one (denoted as *s*) with minimal time cost (Lines 3 and 4). Further, under the cost constraint of maintenance and communication, SRADN_I updates *sr_ent*'s corresponding skyline snapshot from *rootS* to *s* (Lines 5, 6, 7).

We can see from Algorithm 4 that SRADN_I is based on the time cost model, and can preliminarily optimize the set of skyline snapshots.

In Algorithm 5, in order to utilize the core idera of genetic algorithm, SRADN_II constructs a bit vector $V^{(s)}$ for each skyline snapshot *s*. The length of $V^{(s)}$ is the number of subspace skyline recommendation entities in $KY^{(SR)}$. And for each bit *x* in $V^{(s)}$, if the *x*-th subspace skyline recommendation entities processed by *s*, then $V^{(s)}[x]$ equals 1, otherwise equals 0. (Lines 5-11)

In the algorithm, the fitness function $f(\overline{VS})$ evaluates the computation cost of obtaining subspace skyline recommendation from skyline snapshot (Line 10). It is not difficult to see that for a skyline snapshot *s*, the smaller the value of fitness function, the stronger its adaptability. That is, *s* is more excellent.

In Lines 13-23, SRADN_II obtains the new excellent generation set of skyline snapshots by two-point crossover of $\overline{VS}$. While in Lines 24-39, SRADN_II obtains the new excellent generation set of skyline snapshots by two-point mutation of $\overline{VS}$.

Note that for guaranteeing the correctness of the algorithm; in the processes of two-point crossover and two-point mutation, we always let a subspace skyline recommendation only be associated with one skyline snapshot. This can be seen in Line 18 and Lines 27-36, respectively.

It is not difficult to see that SRADN has the polynomial time complexity, which is shown in the following theorem.

- *Theorem 4*. Assume there exists *u* subspace skyline recommendations $SR=\{sr_1, \ldots, sr_u\}$ and γ candidate skyline snapshots $CSN=\{s_1, \ldots, s_\gamma\}$ in the SPA distributed network. Given the user threshold of

maintenance and communication cost *userCost*, and the partition parameter m, the time complexity of SRADN equals (Equation 10):

$$O(\gamma + u + \frac{\gamma u}{m^2} + \frac{\gamma^3}{2m^3}) \quad (10)$$

- *Proof*. The time cost of SRADN mainly includes six parts:

1. $O(\gamma, u)$: the time cost of constructing $KY^{(CSN)}$ and $KY^{(SR)}$ in Algorithm 1.
2. $O(\gamma/m \times u/m)=O(\gamma \times u /m^2)$: in Algorithm 4, for each subspace skyline recommendation, the time cost of obtaining the skyline snapshot with minimal computation cost.
3. $O(\gamma/m \times u/m)=O(\gamma \times u /m^2)$: the time cost of constructing bit vectors in Algorithm 5.
4. $O(\gamma/2m \times (\gamma/m)^2)=O(\gamma^3/2m^3)$: the time cost of two-point crossover in Algorithm 5.
5. $O(\gamma/2m \times \gamma/m)=O(\gamma^2/2m^2)$: the time cost of two-point mutation in Algorithm 5.
6. $O(\gamma/n+u/n+\gamma/n)=O((2\gamma+u)/n)$: the time cost of executing the reduce function in Algorithm 1, where n is the number of computers used to execute the reduce function.

Hence, the time complexity of SRADN equals (Equation 11):

$$\begin{aligned} cost(\text{SRADN}) = &+ O(\gamma \times u/m^2) + O(\gamma \times u/m^2) + O(\gamma^3/2m^3) + \\ &O(\gamma^2/2m^2) + O((2\gamma+u)/n) + O((2\gamma+u)/n) = \\ &O(\gamma+u+\gamma u/m^2+\gamma^3/2m^3) \end{aligned} \quad (11)$$

5. Experimental Evaluation

5.1. Experimental Setting

In our experiments, experimental environment is a three-layer SPA distributed network, which consists of 30 PC. And each PC has a quad-core i5-3450 CPU, 4G memory, 500G hard drive, and CentOS Linux 6.4 operating system.

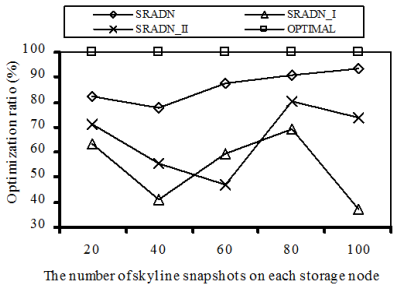
The computation node contains a cluster consisting of 10 PC, in which a PC is selected as the control computer (Master). These 10 PC constitutes a Hadoop platform whose version number is 1.0.3. The remaining two layers include 20 distributed storage nodes, and each node has one PC. In our experiments, we produce 200 subspace skyline recommendations on the computation node, and 100 skyline snapshots on each storage node. Then we totally have 2000 candidate skyline snapshots in the SPA distributed network.

There are three algorithms compared with SRADN:

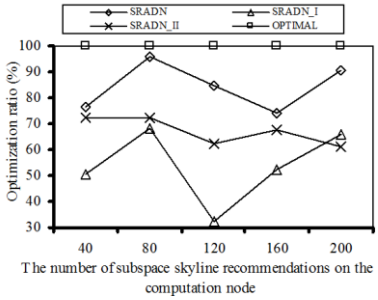
1. OPTIMAL, the algorithm traverses exponential combinations of skyline snapshots to obtain the exact optimal solution.
2. SRADN_I, the algorithm obtains the solution only through the first phase of SRADN.
3. SRADN_II, the algorithm obtains the solution through the second phase of SRADN. Each class of experiments is divided into two groups: the number of subspace skyline recommendations on the computation node is fixed to 100, and the number of skyline snapshots on each storage node varies in the range [20, 100]; and the number of skyline snapshots on each storage node is fixed to 50, and the number of subspace skyline recommendations on the computation node varies in the range [40, 200].

5.2. Performance Evaluation for SRADN

In this subsection, we experimentally evaluate the optimization ratio of SRADN. Figures 1-a and 1-b rspetively show the results of experiments for these four algorithms.



a) The first group of experiments.



b) The second group of experiments.

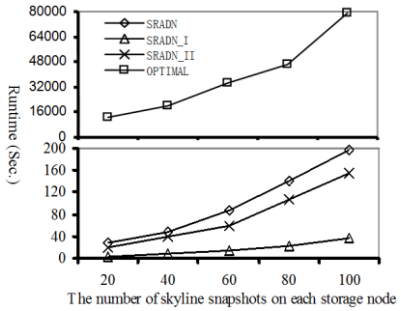
Figure 1. Performance evaluation for four algorithms.

In Figure 1, we take OPTIMAL as the baseline since the set of skyline snapshots produced by OPTIMAL is exactly optimal. And we let the optimization ratio of OPTIMAL equal 100%. From Figure 1, we can observe that the optimization ratio of SRADN approaches the one of OPTIMAL, and the optimization ratios of SRADN_I and SRADN_II are smaller than the one of SRADN. This is mainly because SRADN_I and SRADN_II easily fall into the problem of local optimum, and can not obtain the better set of skyline snapshots. Moreover, we can observe that SRADN_I is the worst one among these four algorithms on optimization ratio. For instance, in Figure 1-a, when the number of skyline snapshots on each storage node equals 100, the optimization ratio of SRADN is equal to 93.6%, while the optimization ratios of SRADN_I and SRADN_II are only 37.5% and 74.1% respectively. And in Figure 1-b, when the number of subspace skyline recommendations on the

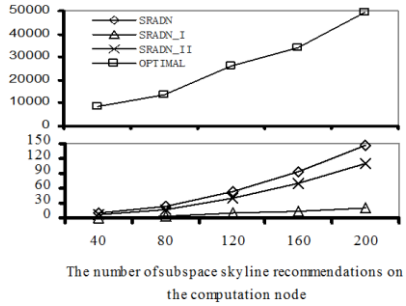
computation node equals 80, the optimization ratio of SRADN is equal to 95.6%, while the optimization ratios of SRADN_I and SRADN_II are only 68.3% and 72.6% respectively.

5.3. Runtime Evaluation for SRADN

In this subsection, we experimentally evaluate the runtime of SRADN. Figures 2-a and b respectively show the results of experiments for these four algorithms.



a) The first group of experiments.



b) The second group of experiments.

Figure 2. Runtime evaluation for four algorithms.

Although, in Figure 1, the optimization ratio of OPTIMAL is slightly higher than the one of SRADN. While in Figure 2, we can see that the runtime of OPTIMAL is huge in each experimental setting. The main reason is:

1. In order to exactly obtain the optimal set of skyline snapshots, OPTIMAL must traverse all possible combinations of skyline snapshots, and has exponential time cost.
2. While SRADN does not need to traverse all possible combinations of skyline snapshots, and only has polynomial time cost to return approximate optimal solution. Moreover, from Figure 2, we can also see that the runtime of SRADN is slight longer than the ones of SRADN_I and SRADN_II, and the runtime of SRADN_I is the shortest among these four algorithms. For instance, in Figure 2-a, when the number of skyline snapshots on each storage node equals 100, the runtime of OPTIMAL equals 79824.6 seconds, while the ones of SRADN, SRADN_I and SRADN_II only is 198.5 seconds, 35.9 seconds and 155.4 seconds respectively. And in Figure 2-b, when the number of subspace skyline recommendations on the computation node equals 200, the runtime of OPTIMAL equals 49652.5 seconds, while the ones of SRADN, SRADN_I and SRADN_II only is 147.8 seconds, 20.4 seconds and 110.4 seconds respectively.

Hence, from the experimental evaluation in Figures 1 and 2, we can get the conclusion that SRADN can efficiently balance the optimization ratio and runtime, and has good extendibility.

7. Conclusions and Future Works

It is very meaningful to research and implement subspace skyline recommendations in SPA distributed networks under the cost constraint of maintenance and communication. In this paper, we analyze the main performance drawbacks of existing works, and propose an efficient algorithm SRADN to efficiently process subspace skyline recommendations in SPA distributed networks. The SRADN algorithm does not need to use fine-grain basic data as the input parameter, and just utilizes prestoring optimal set of skyline snapshots to efficiently process multiple subspace recommendations. Our SRADN algorithm utilizes the map/reduce distributed computation model and can fast produce the optimal set of skyline snapshots through the following two phases: Heuristically constructing the initial set of snapshots, and adjusting the set of snapshots based on the genetic algorithm. The detailed theoretical analyses and extensive experiments demonstrate that our SRADN algorithm is both efficient and effective.

Future work will focus on designing more exact cost evaluation model, improving the processes of two-point crossover and mutation in Algorithm 5, and on more experimentation.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 61272268, 61103069), the Program for New Century Excellent Talents in University (NCET-12-0413), the Fok Ying-Tong Education Foundation (142002).

References

[1] Afrati F. and Ullman J., "Optimizing Multiway Joins in a Map-Reduce Environment," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 9, pp. 1282-1298, 2011.

[2] Borzsonyi S., Kossmann D., and Stocker K., "The Skyline Operator," *in Proceeding of 17th International Conference on Data Engineering*, Heidelberg, pp. 421-430, 2001.

[3] Chaudhuri S., Dalvi N., and Kaushik R., "Robust Cardinality and Cost Estimation for Skyline Operator," *in Proceeding of 22th International Conference on Data Engineering*, Atlanta, pp. 1-

10, 2006.
[4] Chen Q., Zhang Q., and Niu Z., "A Graph Theory based Opportunistic Link Scheduling for Wireless Ad-Hoc Networks," *IEEE Transactions on Wireless Communications*, vol. 8, no. 10, pp. 5075-5085, 2009.
[5] Chomicki J., Godfrey P., Gryz J., and Liang D., "Skyline with Presorting: Theory and Optimization," *in Proceeding of 14th International Conference on Intelligent Information System*, Oslo, pp. 593-602, 2005.
[6] Doulkeridis C., Vlachou A., Nørvåg K., Kotidis Y., and Vazirgiannis M., "Multidimensional Routing Indices for Efficient Distributed Query Processing," *in Proceeding of 18th ACM Conference on Information and Knowledge Management*, Hong Kong, pp. 1489-1492, 2009.
[7] Gasse M., Aussem A., and Elghazel H., "A hybrid Algorithm for Bayesian Network Structure Learning with Application to Multi-Label Learning," *Expert Systems with Applications*, vol. 41, no. 15, pp. 6755-6772, 2014.
[8] Godfrey P., *Skyline Cardinality for Relational Processing*, Springer Berlin Heidelberg, 2004.
[9] Huang Z., Guo J., Sun S., and Wang W., "Efficient Optimization of Multiple Subspace Skyline Queries," *Journal of Computer Science and Technology*, vol. 23, no. 1, pp. 103-111, 2008.
[10] Huang Z., Sun S., and Wang W., "Efficient mining of Skyline Objects in Subspaces Over Data Streams," *Knowledge and Information Systems*, vol. 22, no. 2, pp. 159-183, 2010.
[11] Huang Z., Xiang Y., Sun S., and Chen Q., "Optimizing Skyline Queries in SPA Distributed Networks," *Chinese Journal of Electronics*, vol. 41, no. 8, pp. 1515-1520, 2013.
[12] Huang Z., Xiang Y., Zhang B., and Liu X., "A Clustering based Approach for Skyline Diversity," *Expert Systems with Applications*, vol. 38, no. 7, pp. 7984-7993, 2011.
[13] Hu H., Xu J., Xu X., Pei K., Choi B., and Zhou S., "Private Search on Key-Value Stores with Hierarchical Indexes," *in Proceeding of 30th IEEE International Conference on Data Engineering*, Chicago, pp. 628-639, 2014.
[14] Itmazi J. and Megías M., "Using Recommendation Systems in Course Management Systems to Recommend Learning Objects," *The International Arab Journal for Information Technology*, vol. 5, no. 3, pp. 234-240, 2008.
[15] Pei J., Jiang B., Lin X., and Yuan Y., "Probabilistic Skylines on Uncertain Data," *in Proceeding of 33rd International Conference on Very Large Data Bases*, Vienna, pp. 15-26, 2007.
[16] Pospiech S., Mielke S., Mertens R., Pelke M., Jagannath K., and Stadler M., "Exploration and Analysis of Undocumented Processes using Heterogeneous and Unstructured Business Data," *in Proceeding of International Conference on Semantic Computing*, Newport Beach, pp. 191-198, 2014.
[17] Rodrigo C., Gaspar F., and Lisbona F., "Multigrid Methods on Semi-Structured Grids," *Archives of Computational Methods in Engineering*, vol. 19, no. 4, pp. 499-538, 2012.
[18] Trimponias G., Bartolini I., Papadias D., and Yang Y., "Skyline Processing on Distributed Vertical Decompositions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 4, pp. 850-862, 2013.
[19] Vlachou A., Doulkeridis C., Kotidis Y., and Vazirgiannis M., "SKYPEER: Efficient Subspace Skyline Computation over Distributed Data," *in Proceeding of 23th International Conference on Data Engineering*, Istanbul, pp. 416-425, 2007.
[20] Xu X. and Song M., "Restricted Coverage in Wireless Networks," *in Proceeding of International Conference on Computer Communications*, London, pp. 558-564, 2014.
[21] Zhang N., Li C., Hassan N., Rajasekaran S., and Das G., "On Skyline Groups," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 4, pp. 942-956, 2014.
[22] Zhang Q., Fu H., and Qiu G., "Tree Partition Voting Min-Hash for Partial Duplicate Image Discovery," *in Proceeding of International Conference on Multimedia and Expo*, San Jose, pp. 1-6, 2013.
[23] Zhang W., Zhang S., Qi F., and Cai M., "Self-Organized P2P Approach to Manufacturing Service Discovery for Cross-Enterprise Collaboration," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 44, no. 3, pp. 263-276, 2014.

Zhenhua Huang is currently an associate professor at the School of Electronics and Information, Tongji University. He received his PhD. degree in computer science from Fudan University. His research interests include information service, data mining and knowledge discovery. He has published over 50 papers in various journals and conference proceedings.

Jiawen Zhang received her BSc degree in computer science from Tongji University. She is currently a MSc student at Tongji University. She has authored a number of journal and conference papers in the fields of data mining, query optimization and information recommendation.