EFFICIENT OPTIMIZATION FOR L-EXTSKY RECOMMENDATIONS

Zhenhua Huang, Juru Wang, Bo Zhang

L-extendSKY recommendation has recently received a lot of attention in information retrieval community. Literature [1] proposes an algorithm EARG (Efficient Approach based on Regular Grid) to produce the L-extendSKY objects in one single subspace. However, in multi-user environments, the system generally handles multiple subspace L-extendSKY recommendations simultaneously. Hence, in this paper, we present an efficient algorithm AOMSR (Algorithm for Optimizing Multiple Subspace L-extendSKY Recommendations) to remarkably reduce the total response time. Furthermore, we discuss two interesting variations of L-extendSKY recommendation, i.e., global constraint L-extendSKY recommendation and local constraint L-extendSKY recommendation, which are meaningful in practice, and show how our algorithm can be applied for their efficient processing. Detailed theoretical analyses and extensive experiments that demonstrate our solution are both efficient and effective.

Keywords: information retrieval; L-extendSKY recommendation; subspace; performance evaluation

1 Introduction

The skyline recommendation and its computation have attracted much attention recently [2]. To my best knowledge, various techniques have been proposed for subspace skyline recommendation. The existing approaches can be classified in three categories: (1) The first category [3, 4] involves the solutions that assume that the recommendation subspace is fixed and use different index structures to improve recommendation performance. (2) Methods in the second category [5, 6] consider how to efficiently process all \( 2^k-1 \) subspace skyline recommendations. (3) The third category [7, 8] tackles the problem of optimizing arbitrary single subspace skyline recommendations. Clearly, if the input dataset is fixed, then the recommendation results returned by these existing approaches will keep invariant.

Literature [1] points out that in most real applications, for a \( \zeta \)-dimensional dataset \( AD \), the cardinality of its recommendation result does not exceed \( \left( \frac{\ln \zeta}{1} \cdot |AD| \right) / \left( |AD| \cdot (\zeta-1)! \right) \) of that of \( AD \) [9]. So, the recommendation result returned by the existing approaches cannot efficiently assist the users to explore the whole dataset. Motivated by the above fact, the literature [1] extends the semantics of skyline recommendation and proposes a new type of recommendation which is called L-extendSKY recommendation. Given a set of \( \zeta \)-dimensional objects, a L-extendSKY recommendation on the subspace \( V \) \( |V| \leq \zeta \) finds the objects that are dominated by at most \( L \) objects on \( V \). Conceptually, \( K \) represents the thickness of the skylines; the special case \( K = 0 \) corresponds to the conventional skyline recommendation. It is easy to see that compared with the traditional skyline recommendation, the L-extendSKY recommendation has at least 2 advantages: (1) the L-extendSKY recommendation can provide more opportunities for users to explore the whole input dataset, since it also considers the non-skyline objects; and (2) the users can flexibly adjust the parameter \( L \) to obtain the recommendation result which they need. Consequently, the L-extendSKY recommendation is more meaningful in practice. Furthermore, the literature [1] presents an algorithm EARG (Efficient Approach based on Regular Grid) to produce the L-extendSKY objects in arbitrary single subspace. The EARG approach utilizes the regular grid structure and prunes all the cells which are dominated by any other ones, and hence it can evidently reduce the number of comparisons between objects.

However, in multi-user environments, the system generally handles multiple subspace L-extendSKY recommendations simultaneously. Hence, in this paper, we propose an efficient algorithm AOMSR (Algorithm for Optimizing Multiple Subspace L-extendSKY Recommendations) to markedly reduce the total recommendation time. The AOMSR algorithm first organizes all issued subspace L-extendSKY recommendations as a recommendation tree, and then uses the share mechanism in tree paths to improve the total performance of these L-extendSKY recommendations. Moreover, we discuss two interesting variations of L-extendSKY recommendation, i.e., global constraint L-extendSKY recommendation and local constraint L-extendSKY recommendation, which are meaningful in practice, and show how our algorithm can be applied for their efficient processing. Detailed theoretical analyses and extensive experiments that demonstrate our solution are both efficient and effective.
2 The EARG approach

In this section, we briefly review the EARG approach proposed in the literature [1].

The EARG approach uses the regular grid [10] to index the objects. Consider a set \( AD \) of objects which totally has \( \zeta \) dimensions. And each dimension \( d \) over \( AD \) has a set of disjoint ranges which partition the value domain of \( d \). Let the extent of each cell on the dimension \( d \) (\( 1 \leq i \leq \zeta \)) be \( \delta _i \). Then the cell \( c[a_1, \ldots, a_\zeta] \) contains all objects with the \( i \)-th dimension in the range \( [a_i - (\delta_i -1), a_i \cdot \delta_i) \). Conversely, give an object \( p \) with attributes \( (p.x_1, \ldots, p.x_\zeta) \), its covering cell can be determined (in constant time) as \( c[b_1, \ldots, b_\zeta] \) where \( b_i(1 \leq i \leq \zeta) = \lfloor p.x_i / \delta_i \rfloor \). Moreover, the EARG approach uses the following four definitions and one theorem, which provide us an opportunity to optimize the performance of the L-extSKY recommendations.

**Definition 1.** Assume that the full space \( F \) consists of \( \zeta \) dimensions \( \{d_1, \ldots, d_\zeta\} \), and the subspace \( V \) consists of \( \nu \leq \zeta \) dimensions \( \{d_1, \ldots, d_\nu\} \). Then if a cell \( C(V) = [a_1, \ldots, a_\nu] \) on \( V \) satisfies the following condition, we call \( C(V) \) the \( \nu \)-dimensional cell of the cell \( C(F) = [b_1, \ldots, b_\nu, \ldots, b_\zeta] \) on \( F \): \( \forall i \in [1, \nu], a_i < b_i \).

In order to efficiently realize the EARG approach, the literature [1] distinguishes three possibilities about the relationship between any two \( \nu \)-dimensional cells \( C_1(V) = [a_1, \ldots, a_\nu] \) and \( C_2(V) = [b_1, \ldots, b_\nu] \), satisfying the condition, then we say “\( C_1(V) \) fully dominates \( C_2(V) \)”:

- For simplicity, literature [1] denotes this relationship as \( C_1(V) <_{\nu} C_2(V) \).
- If two \( \nu \)-dimensional cells \( C_1(V) = [a_1, \ldots, a_\nu] \) and \( C_2(V) = [b_1, \ldots, b_\nu] \) satisfy the following condition, then we say “\( C_1(V) \) partially dominates \( C_2(V) \)”:
  \[ \exists U \subseteq \nu \forall i \in [1, \nu], a_i < b_i \text{ and } \forall j \in ([U], \nu), a_j < b_j. \]

For simplicity, literature [1] denotes this relationship as \( C_1(V) \approx_{\nu} C_2(V) \).
- If two \( \nu \)-dimensional cells \( C_1(V) = [a_1, \ldots, a_\nu] \) and \( C_2(V) = [b_1, \ldots, b_\nu] \) satisfy the following condition, then we say “\( C_1(V) \) is incomparable with \( C_2(V) \)”:
  \[ \exists i \in [1, \nu], a_i < b_i \text{ and } \exists j \in [1, \nu], b_j < a_j. \]

For simplicity, literature [1] denotes this relationship as \( C_1(V) \succ_{\nu} C_2(V) \).

**Theorem 1.** Assume that two \( \nu \)-dimensional cells \( C_1(V) = [a_1, \ldots, a_\nu] \) and \( C_2(V) = [b_1, \ldots, b_\nu] \) cover the objects sets \( S_1(V) \) and \( S_2(V) \), respectively. Then we can have:

- \( C_1(V) <_{\nu} C_2(V) \) \( \implies \forall p \in S_1(V), \forall r \in S_1(V), r \) is dominated by \( p \);
- \( C_1(V) \approx_{\nu} C_2(V) \) \( \implies \forall p \in S_1(V), \forall r \in S_1(V), p \) is not dominated by \( r \);
- \( C_1(V) \succ_{\nu} C_2(V) \) \( \implies \forall p \in S_1(V), \forall r \in S_1(V), p \) is not dominated by \( r \).

Based on the definitions 1-4 and the theorem 1, the EARG approach can be efficiently implemented in the algorithms 1 and 2.

**Algorithm 1: EARG**

**Input:** the set of \( \zeta \)-dimensional objects \( AD \), each object is associated with one counter \( count \) whose initial value equals 0; the subspace \( V \) and its dimensionality \( \nu \); the regular grid index \( \mathcal{X}(AD, F) \); the parameter \( L \).

**Output:** the \( L \)-extSKY set \( L-\text{EXSET}(AD, V) \).

**Begin**

1. \( L-\text{EXSET}(AD, V) \leftarrow \emptyset; \)
2. Obtain all \( V \)-dimensional cells which contain at least one object from \( \mathcal{X}(AD, F) \);
3. Organize these non-empty cells as a sequence \( Seq \) which satisfies: for any two cells \( C(V) = [a_1, \ldots, a_\nu] \) and \( C(V) = [b_1, \ldots, b_\nu] \), if \( a \prec \beta \), then \( \sum_{i=1}^{\nu} a_i \leq \sum_{i=1}^{\nu} b_i \)
4. For orderly visit each cell \( C(V) \) in \( Seq \) Do
5. \( S(V) \leftarrow \text{the set of objects inside } C(V) ; \)
6. \( CR(C(V)) \leftarrow \text{OBCANS}(S(V), V, L) \);
7. \( flag \leftarrow \text{False}; \)
8. For orderly visit each cell \( C(V) \) in \( Seq \) which \( Locate \) Do
9. \( IF C(V) \leftarrow \text{OBCANS Then} \)
10. \( \forall p \in CR(C(V)) \) Do
11. \( count \leftarrow p . count + |S(V)| ; \)
12. \( IF p . count \geq L \) Then \( flag \leftarrow \text{True}; \)
13. \( CR(C(V)) \leftarrow \text{CR}(C(V)) \times \{p\} ; \)
14. \( IF C(V) <_{\nu} C_2(V) \) Then
15. \( \forall p \in CR(C(V)), \forall r \in S(V) \) Do
16. \( IF p . count \neq r \) Then
17. \( p . count \leftarrow p . count + 1 ; \)
18. \( IF p . count \geq L \) Then \( flag \leftarrow \text{True}; \)
19. \( CR(C(V)) \leftarrow CR(C(V)) \times \{p\} ; \)
20. \( IF flag = \text{True} \) Then \( Seq \leftarrow Seq \times CR(C(V)) \); \( C(\nu) <_{\nu} C_\nu \);
21. \( L-\text{EXSET}(AD, V) \leftarrow L-\text{EXSET}(AD, V) \cup CR(C(V)); \)
22. \( Return L-\text{EXSET}(AD, V) \);

**End**

In the step 3 of Algorithm 1, any two cells \( C(V) = [a_1, \ldots, a_\nu] \) and \( C(V) = [b_1, \ldots, b_\nu] \) in \( Seq \) need to satisfy the condition: \( a \prec \beta \), which has at least two advantages: (1) If a cell \( C(V) \) is visited earlier, then the probability that it is fully or partially dominated by other cells is lower. (2) If a cell \( C(V) \) locates before another cell \( C(V) \), then \( C(V) \) cannot fully or partially dominate \( C(V) \), and hence according to Theorem 1, we can have that the objects in \( S(V) \) cannot dominate any object in \( S(V) \) on \( V \). Consequently, this condition can remarkably reduce the number of comparisons between objects. For each visited cell \( C(V) \), the step 6 uses the function OBCANS which is described in Algorithm 2 to obtain all candidate L-extSKY objects in \( S(V) \); that is, if \( p \in CR(C(V)) \), then \( p \) is dominated by at most \( k \) objects in \( S(V) \) on \( V \). In the steps 8+19, for each object \( p \in CR(C(V)) \), the algorithm modifies its counter. And if \( p . count \) exceeds \( L \) then \( p \) can be safely removed from \( CR(C(V)) \). Specially, the steps 9+13 focus on considering each cell \( C(V) \) which fully dominates \( C(V) \), while the steps 14+19 focus on considering each cell \( C(V) \) which partially dominates \( C(V) \). In the steps 9+13, if \( C(V) \) fully dominates \( C(V) \), then according to Definition 2, for
each object \( p \in CR(C_\alpha(V)) \), \( p \) does not need to be compared with any objects in \( C_\alpha(V) \), and \( p\text{-count} \) is directly added to \( |S_\alpha(V)| \). While in the steps 14+19, if \( C_\alpha(V) \) partially dominates \( C_\alpha(V) \), then according to Definition 3, for each object \( p \in CR(C_\alpha(V)) \), \( p \) needs to be compared with all objects in \( S_\alpha(V) \), and \( p\text{-count} \) is added the number of objects in \( S_\alpha(V) \) which dominate \( p \) on \( V \). Furthermore, in the step 20, if the algorithm finds that there exists some object \( p \in S_\alpha(V) \) whose counter is greater than or equal to \( L \), then we can safely remove all cells from \( seqV \) which are fully dominated by \( C_\alpha(V) \).

The function OBCANS utilizes the thought of presorting objects in [4] to obtain \( CR(C_\alpha(V)) \), which can be shown in the following algorithm.

**Algorithm 2: OBCANS**

**Input:** the set \( S_\alpha(V) \) that consists of all objects inside \( C_\alpha(V) \); the subspace \( V \); the parameter \( L \).

**Output:** the set \( CR(C_\alpha(V)) \) that consists of all candidate \( L\text{-extSKY} \) objects in \( S_\alpha(V) \);

**Begin**

1. \( CR(C_\alpha(V)) \leftarrow \emptyset \);
2. For each object \( p \in S_\alpha(V) \) Do \( p\text{-key} \leftarrow \Sigma_i \ln(p[i] + 1) \);
3. \( L_\alpha(V) \leftarrow \) the object list obtained by sorting all the objects in \( S_\alpha(V) \) in the key ascending order;
4. \( flag_\alpha \leftarrow \) False;
5. For orderly visit each object \( p \in L_\alpha(V) \) Do
6. \( flag_\alpha \leftarrow \) False;
7. For orderly visit each object \( r \in L_\alpha(V) \) which locates before \( p \) Do
8. If \( p \) dominates \( r \) Then \( p\text{-count} \leftarrow p\text{-count} + 1 \);
9. If \( p\text{-count} \geq L \) Then \( flag_\alpha \leftarrow \) True;
10. If \( p\text{-count} > L \) Then \( flag_\alpha \leftarrow \) True; break;
12. If \( flag_\alpha \) \leftarrow \) True Then \( CR(C_\alpha(V)) \leftarrow CR(C_\alpha(V)) \cup \{ p \} \);
13. If \( flag_\alpha \) \leftarrow \) True Then \( seqV \leftarrow seqV \cup \{ C_\alpha(V) | C_\alpha(V) \lessdot \alpha \in CR(C_\alpha(V)) \} \);
15. Return \( CR(C_\alpha(V)) \);

In the steps 2 to 3 of Algorithm 2, we sort the objects in \( S_\alpha(V) \) using the sorting-operator \( \Sigma_i \ln(p[i] + 1) \), which has at least three advantages: (1) If an object \( p \) in \( L_\alpha(V) \) is visited earlier, then the probability that it is dominated by other objects in \( L_\alpha(V) \) on \( V \) is lower. (2) The probability that different objects in \( L_\alpha(V) \) have the same value of \( \Sigma_i \ln(p[i] + 1) \) is lower. (3) It ensures that any object in \( L_\alpha(V) \) cannot be dominated by other objects located after it on \( V \), and hence can dramatically reduce the number of comparisons between objects. For each object \( p \in L_\alpha(V) \), in the steps 7-12, the algorithm modifies \( p\text{-count} \). And if \( p\text{-count} \) does not exceed \( L \), then the algorithm adds \( p \) into \( CR(C_\alpha(V)) \). Furthermore, in the algorithm, we use the Boolean variant \( flag_\alpha \) as a cell-pruning indicator. When \( flag_\alpha \) \leftarrow True, we can know that there exists some object in \( CR(C_\alpha(V)) \) whose counter is greater than or equal to \( L \). And hence we can safely remove all cells from \( seqV \) which are fully dominated by \( C_\alpha(V) \).

3 Optimizing multiple subspace \( L\text{-extSKY} \) recommendations simultaneously

This section focuses on optimizing multiple subspace \( L\text{-extSKY} \) recommendations simultaneously in multi-user environments. Let \( u \) be the number of subspace \( L\text{-extSKY} \) recommendations handled in multi-user environments. A naïve solution is to run the EARG approach [1] on the original datasets \( u \) times. Obviously, the naïve solution becomes inefficient as the cardinality of original datasets increases, which can be seen in our experimental evaluation. Motivated by this fact, we propose an efficient algorithm AOMSR to reduce the total recommendation time. The AOMSR algorithm first organizes these \( u \) \( L\text{-extSKY} \) recommendations as a subspace recommendation tree, and then exploits the share mechanism in tree paths to reduce the total recommendation time. For easily understanding, in the following parts, we use \( p\text{-count} \) to denote the counter of \( p \) on \( V \).

**Definition 5.** Assume that the list \( SQB \) contains \( u \) subspaces \( <V_0, \ldots, V_u> \). Then if \( SQB \) satisfies the following two properties, then we call it a consistent subspace sequence: (1) \( V_0 = \bigcup_{i=1}^{u} V_i \); and (2) \( \forall V_p, V_j \in SQB, i < j \Rightarrow |V_p| \leq |V_j| \), where \(|V|\) is the dimensionality of \( V \).

**Definition 6.** The subspace recommendation tree \( T_{sb} = (ND, ES) \) is built over a consistent subspace sequence \( SQB = <V_0, \ldots, V_u> \) and needs to satisfy the following properties.

- **Property 1 (of ND).**
  - \( V_0 \) is the root of \( T_{sb} \);
  - \( ND = \{ V_j \mid J \in SQB \} \);
- **Property 2 (of \( <V_p, V_j> \in ES)\).**
  - \( V_p \subset V_j \);
  - \( \exists V_j \in ND, V_j \supseteq V_p \land V_j \supseteq V_j \);
  - \( \exists V_j \in ND, such that <V_p, V_j> \in ES \) and \( <V_p, V_j> \) satisfies Property 2 (a) and (b), and \( h < i \).

**Definition 7.** Let \( AD \) be the set of \( \zeta \)-dimensional objects, and \( rt \) be the root node of the subspace recommendation tree \( T_{sb} \). And for each non-root node \( V \), we assume that \( PV \) is its parent node. Then we can recursively define the seed subspace \( L\text{-extSKY} \) set for each node in \( T_{sb} \) as follows:

\[
\text{seed}(V) = \begin{cases} 
L - \text{EXSET}(AD, rt) & \text{iff } V = rt; \\
L - \text{EXSET}(\text{seed}(PV), V) & \text{otherwise}.
\end{cases}
\]

**Definition 8.** Let \( AD \) be the set of \( \zeta \)-dimensional objects, and \( rt \) be the root node of the subspace recommendation tree \( T_{sb} \). And for each non-root node \( V \), we assume that \( PV \) is its parent node. Then based on the seed subspace \( L\text{-extSKY} \) set, we can define the shadow subspace \( L\text{-extSKY} \) set for each node in \( T_{sb} \) as follows:

\[
rep(V) = \begin{cases} 
\emptyset & \text{iff } V = rt, \\
\{ p \mid p \in AD - \text{seed}(PV) \land \exists r \in \text{seed}(V), \\
\forall z \in V, p[z] = r[z] \} & \text{otherwise}.
\end{cases}
\]

**Theorem 2.** Let \( AD \) be the set of \( \zeta \)-dimensional objects, and \( rt \) be the root node of the subspace recommendation tree \( T_{sb} \). And for each non-root node \( V \), we assume
that $PV$ is its parent node. Then for each node $V$ in $T_{ab}$, we can have:

$L-\text{EXSET}(AD, V) = L-\text{EXSET}(\text{seed}(V) \cup \text{rep}(V), V)$. 

**Proof:** Assume the root node $rt$ is at level 1 in the tree $T_{ab}$. Then, for $rt$, we can have: $L-\text{EXSET}(\text{seed}(rt) \cup \text{rep}(rt), rt) = L-\text{EXSET}(L-\text{EXSET}(AD, rt) \cup \emptyset, rt) = L-\text{EXSET}(AD, rt)$. So, the theorem holds for the root node $rt$. Below, we focus on each non-root node $V$ and prove that: (1) $L-\text{EXSET}(AD, V) \subseteq L-\text{EXSET}(\text{seed}(V) \cup \text{rep}(V), V)$, and (2) $L-\text{EXSET}(\text{seed}(V) \cup \text{rep}(V), V) \subseteq L-\text{EXSET}(AD, V)$. For (1), we prove its correctness by contradiction. Assume there exists an object $p$ such that $p \notin L-\text{EXSET}(AD, V)$ and $p \notin L-\text{EXSET}(\text{seed}(V) \cup \text{rep}(V), V)$. Since $p \notin L-\text{EXSET}(\text{seed}(V) \cup \text{rep}(V), V)$, there exists a set $\mathcal{J}$ such that $|\mathcal{J}| > L$ and $\forall r \in \mathcal{J}$, $p$ dominates $r$. And since $\text{seed}(V) \cup \text{rep}(V) \subseteq AD$, $p \notin L-\text{EXSET}(AD, V)$. This contradicts the assumption that $p \notin L-\text{EXSET}(AD, V)$. Therefore, $L-\text{EXSET}(AD, V) \subseteq L-\text{EXSET}(\text{seed}(V) \cup \text{rep}(V), V)$. For (2), we also prove its correctness by contradiction. Assume there exists an object $p$ such that $p \notin L-\text{EXSET}(\text{seed}(V) \cup \text{rep}(V), V)$ and $p \notin L-\text{EXSET}(AD, V)$. From the assumption, we can easily see that there exists some object $q$ such that $q \in AD - (\text{seed}(V) \cup \text{rep}(V))$ and $p$ dominates $q$. Since $q \notin \text{seed}(V) \cup \text{rep}(V)$, $q \notin L-\text{EXSET}(\text{seed}(V) \cup \text{rep}(V), V)$. Hence, $\text{count}_V > L$. Therefore, we can have: $p.\text{count}_V > L$. That is, $p \in \text{L-EXSET}(\text{seed}(V) \cup \text{rep}(V), V)$. This contradicts the above assumption that $p \notin L-\text{EXSET}(\text{seed}(V) \cup \text{rep}(V), V)$. So then, $L-\text{EXSET}(\text{seed}(V) \cup \text{rep}(V), V) \subseteq L-\text{EXSET}(AD, V)$. Based on the above analyses, we can know that the theorem holds.

The AOMSR algorithm is based on Theorem 2 and can be shown below.

**Algorithm 3: AOMSR**

**Input:** the parameter $L$; the set of $\zeta$-dimensional objects $AD$; the regular grid index $\mathcal{X}(AD, F)$; $u$ L-estSKY recommendations whose corresponding subspaces are $V_1, \ldots, V_u$.

**Output:** $u$ subspace L-estSKY sets $L-\text{EXSET}(AD, V_1), \ldots, L-\text{EXSET}(AD, V_u)$.

**Begin**

1. $\text{SQB} \leftarrow$ the consistent subspace sequence containing $u$ subspaces;
2. $T_{ab} \leftarrow$ the subspace recommendation tree over $\text{SQB}$;
3. For each node $V$ in $T_{ab}$ Do
4. If $V$ is the root node $rt$ Then
5. $L-\text{EXSET}(AD, rt) \leftarrow E\text{ARG}(AD, rt, \mathcal{X}(AD, F), L)$;
6. Else
7. $PV$ the parent node of $V$;
8. $\text{seed}(V) \leftarrow E\text{ARG}(\text{seed}(PV), V, \mathcal{X}(AD, F), L)$;
9. $\Delta(V) \leftarrow \text{seed}(V)$;
10. Divide $\text{seed}(V)$ into $m$ subsets $SD_1, \ldots, SD_m$ which satisfy the following conditions:
(a) $\forall p, r \in SD_i, \forall z \in V, p[z] = r[z]$,
(b) $\forall p, r \in SD_i, r[z] \geq 3, z \in V, p[z] \neq r[z]$;
11. For each $SD_i$ Do
12. Select some object $p$ from $SD_i$;
13. $SD_i.\text{key} \leftarrow \sum_{z \in V} p[z]$;
14. Organize these $m$ subsets as a list $LD = <SD_1, \ldots, SD_m>$ which satisfies: $i < t \Rightarrow SD_i.\text{key} \leq SD_t.\text{key}$;
15. For each subset $SD_i$ in $LD$ Do
16. Select some object $p$ from $SD_i$;
17. $SD_i \leftarrow \{q \in AD - \Delta(PV) \mid q$ falls inside the covering cell of $p$, $\exists z \in V, p[z] = q[z] \}$;
18. For each object $q \in SD_i$ Do $\text{count}_q \leftarrow p.\text{count}_i$;
19. For each $SD_i$ which locates after $SD_j$ in $LD$ Do
20. Select some object $r$ from $SD_i$;
21. If $r$ dominates $p$ Then
22. if $r.\text{count}_i > |SD_j|$ Then
23. $\text{seed}(V) \leftarrow \text{seed}(V) - SD_j$;
24. $LD \leftarrow LD - |SD_j|$;
25. Else For each object $\delta \in SD_j$ Do $\delta.\text{count}_i \leftarrow \delta.\text{count}_i + |SD_j|$
26. $\text{rep}(V) \leftarrow \text{rep}(V) - SD_j$;
27. $L-\text{EXSET}(AD, V) \leftarrow \text{seed}(V) \cup \text{rep}(V)$;
28. Return $u$ subspace $L-\text{estSKY}$ sets $L-\text{EXSET}(AD, V_1), \ldots, L-\text{EXSET}(AD, V_u)$.

**End**

In Algorithm 3, after obtaining the subspace recommendation tree $T_{ab}$ (Steps 1+2), the AOMSR algorithm visits each node $V$ and obtains its subspace L-estSKY set $L-\text{EXSET}(AD, V)$ in a breadth-first mode (steps 3+28). In the steps 4 and 5, for the root node $rt$, since its seed subspace L-estSKY set is directly obtained from the original dataset $AD$ using the EARG approach, $\text{seed}(rt) = L-\text{EXSET}(AD, rt)$. While for each non-root node $V$, the AOMSR algorithm needs to utilize the property of Theorem 3 to obtain $L-\text{EXSET}(AD, V)$. The basic idea is that the algorithm first obtains the seed subspace L-estSKY set $\text{seed}(V)$ in step 8, and then uses $\text{seed}(V)$ to produce the correct subspace L-estSKY set $L-\text{EXSET}(AD, V)$ in the succeeding steps. In the step 10, the algorithm divides $\text{seed}(V)$ into $m$ subsets and all the objects in the same subsets share the same values on $V$. And in the steps 11+13, for each subset $SD_i$, the algorithm obtains its key $SD_i.\text{key}$. Then the step 14 organizes these $m$ subsets as a list $LD = <SD_1, \ldots, SD_m>$ satisfying: $i < t \Rightarrow LD[i].\text{key} \leq LD[t].\text{key}$.

Note that for any two subsets $SD_i$ and $SD_j$ in $LD$, if $SD_j$ locates before $SD_i$, then the objects in $SD_j$ cannot dominate the objects in $SD_i$ on $V$. After obtaining the list $LD$, the algorithm orderly processes each subset $SD_i$. In the steps 16+17, the algorithm first randomly selects an object $p$ from $SD_i$ and obtains the set $SB_i$ which consists of the objects that belong to $AD - \Delta(PV)$ and fall inside the covering cell of $p$ and share the same values on $V$ with $p$. Then in the step 18, for each object $q \in SB_i$, the algorithm set $q.\text{count}_i$ to be equal to $p.\text{count}_i$. Since the objects in $SB_i$ are added into the shadow subspace L-estSKY set $\text{rep}(V)$ and ultimately are incorporated into $L-\text{EXSET}(AD, V)$ (steps 27+28), the algorithm needs to handle each subset $SD_i$ which locates after $SD_j$ in $LD$ (steps 19+26).

The handling process can be described as follows. The algorithm first randomly selects an object $r$ from $SD_j$ and checks if $r$ is dominated by $p$ on $V$. If yes, then the algorithm continues to evaluate the value of $r.\text{count}_i + |SB_j|$. If $r.\text{count}_i + |SB_j|$ is greater than $K$, then the algorithm deletes all the objects in $SD_i$ from $\text{seed}(V)$ and removes $SD_j$ from $LD$; otherwise, for each object $\delta$ in $SD_i$, $\delta.\text{count}_i$ is added $|SB_j|$. 

1102  Technical Gazette 22, 5(2015), 1099-1106
Example 1. Assume there exists a set of 2-dimensional objects \( AD = \{ p_1(0.5, 4.9), p_2(2.2, 3.1), p_3(9.7, 2.8), p_4(3.9, 3.3), p_5(5.6, 2.4), p_6(4.9, 2.1), p_7(6.5, 4.9), p_8(3.7, 3.7), p_9(9.1, 2.4) \} \). And the user Jukie needs two 1-EXTSKY recommendations whose corresponding subspaces are \( V^1 = \{ d_1, d_2 \} \) and \( V^2 = \{ d_2 \} \). The AOMSR algorithm first obtains the subspace recommendation tree \( T_{AOM} : V^1 \rightarrow V^2 \) and then orderly processes these two nodes. For \( V^1 \), the AOMSR algorithm invokes the EARG approach to obtain \( seed(V_1) \) from \( AD \). And \( 1 \)-EXTSET\((AD, V^1) = \{ p_1, p_2, p_3, p_4, p_5 \} \). For \( V^2 \), the AOMSR algorithm first invokes the EARG approach to obtain \( seed(V_2) \) from \( seed(V_1) \), and \( seed(V^2) = \{ p_3, p_4 \} \). Then, the AOMSR algorithm divides \( seed(V_2) \) into two subsets \( SD_1 = \{ p_3 \} \) and \( SD_2 = \{ p_4 \} \), and organizes these two subsets as a list \( LD = SD_2 \subseteq SD_1 \). The AOMSR algorithm orderly handles these two subsets. For \( SD_2 \), the AOMSR algorithm first randomly selects an object (i.e., \( p_3 \)) from \( SD_2 \) and then obtains the set \( SB_{rep} = \emptyset \). Since \( SB_{rep} \) does not contain any object, the AOMSR algorithm is not necessary to process \( SD_2 \) further. For \( SD_1 \), the AOMSR algorithm also first randomly selects an object (i.e., \( p_4 \)) from \( SD_1 \) and then obtains the set \( SB_{rep} = \{ p_3 \} \). Then, the AOMSR algorithm sets \( ps = count_{p_3} = 1 \) and adds \( p_3 \) into \( rep(V_2) \). Hence, \( 1 \)-EXTSET\((AD, V^1) = seed(V_1) \cup rep(V_2) = \{ p_3, p_4, p_5 \} \).

4 Variations of subspace L-EXTSKY recommendation

In many real applications, users are only interested in some special data space defined by the constraints. Hence, we only need to return the subspace L-EXTSKY set \( L \)-EXTSET\((AD, V) \) within this data space. Typically, each constraint is expressed as a range along a dimension and the conjunction of all constraints forms a hyperrectangle in the \( n \)-dimensional attribute space. Clearly, according to the practical requirements of users, there are two types of recommendations that users might need, which are respectively described in Definitions 9 and 10.

Definition 9. Let \( AD \) be the set of \( \zeta \)-dimensional objects, and \( V = \{ d_1, …, d_j \} \) be the issued subspace. And for each \( d_i \in V \), users are only interested in the region \( \lambda_i = [B_i, E_i] \). Then the global constraint L-EXTSKY recommendation on \( V \) returns the set \( L \)-EXTSET\((AD, V, \lambda) = L \)-EXTSET\((SD, V, \lambda) \), where \( SD = \{ p \in AD \wedge \forall i \in [1, v], B_i \leq p[i] \leq E_i \} \) and \( \lambda = \lambda_1 \wedge … \wedge \lambda_n \).

From Definition 10, we can see that \( L \)-EXTSET\((AD, V, \lambda) \) is the subspace L-EXTSKY set which is obtained from the set \( SD \) consists of the objects which fall inside the data space defined by the constraints \( \lambda \). It is not difficult to see that \( L \)-EXTSET\((AD, V, \lambda) \subseteq L \)-EXTSET\((SD, V, \lambda) \). For simplicity, we call \( L \)-EXTSET\((AD, V, \lambda) \) local constraint subspace L-ExtSKY set.

Example 3. Returning to Example 1, \( 1 \)-EXTSET\((AD, V^1) = \{ p_1, p_2, p_3, p_4, p_5 \} \). When the user Jukie gives the constraints \( \lambda = \lambda_1 \wedge \lambda_2 = \lambda_3 \wedge \lambda_4 \wedge \lambda_5 \) where \( \lambda_1 = [2, 4] \) and \( \lambda_2 = [3, 4] \). Then, \( 1 \)-EXTSET\((AD, V^1) = \{ p_2, p_3, p_4 \} \).
nates \( r \Rightarrow \lambda (r) \Rightarrow - (\lambda (p) \wedge (p \text{ dominates } r)) \lor \lambda (r); \exists \ p, r \in AD, \lambda (p) \wedge (p \text{ dominates } r) \Rightarrow \lambda (r). \) Hence, \( \exists \ p, r \in L-\text{EXSET}(\lambda (AD), V_0^{(0)}) \Rightarrow \exists \ p, r \in \lambda (L-\text{EXSET}(AD), V_0^{(0)}). \) This contradicts with the above assumption that \( \lambda (L-\text{EXSET}(AD), V_0^{(0)}) = L-\text{EXSET}(\lambda (AD), V_0^{(0)}). \) Hence, the theorem stands.

(2) Inductive Hypothesis. For \( z \leq n, \forall p, r \in AD, \lambda (p) \wedge (p \text{ dominates } r) \Rightarrow \lambda (r) \Rightarrow \lambda (\lambda (L-\text{EXSET}(AD), V_0^{(0)})) = L-\text{EXSET}(\lambda (AD), V_0^{(0)}). \) According to the hypothesis: for \( z \leq n, \forall p, r \in AD, \lambda (p) \wedge (p \text{ dominates } r) \Rightarrow \lambda (r) \Rightarrow \lambda (L-\text{EXSET}(AD), V_0^{(0)}) = L-\text{EXSET}(\lambda (AD), V_0^{(0)}). \) Hence, \( \lambda (p) \wedge (p \text{ dominates } r) \Rightarrow \lambda (r) \Rightarrow \lambda (L-\text{EXSET}(AD), V_0^{(0)}) \Rightarrow L-\text{EXSET}(\lambda (AD), V_0^{(0)}). \) That is, \( \lambda (p) \wedge (p \text{ dominates } r) \Rightarrow \lambda (r) \Rightarrow \lambda (L-\text{EXSET}(AD), V_0^{(0)}) = L-\text{EXSET}(\lambda (AD), V_0^{(0)}). \)

5 Experimental evaluation

This section conducts an empirical study of our methods using the benchmark synthetic datasets. We evaluate the efficiency and the scalability of the proposed methods. Using the data generator [4], we generate two types of synthetic datasets as described in [4]: (a) Independent datasets where the dimension values of the generated objects are uniformly distributed; (b) Anti-correlated datasets where if an object is good in one dimension, it is unlikely to be good in other dimensions. Each object totally has eight dimensions whose data types are 4-type float. Furthermore, in the following experiments, we fix the number of ranges over each dimension to 5. All our experiments are implemented in Java, running on a PC with i5-3210M 2.50 GHz processor and 2 G main memory.

5.1 Evaluating multiple subspace L-EXTSKY recommendations

In this subsection, we focus on evaluating the efficiency of our AOMSR algorithm. It is important to note that in order to show the superiority of the AOMSR algorithm, we also evaluate the naïve solution (i.e., running the EARG approach \( u \) times) in the experiments where \( u \) is the number of subspaces. We implement this set of experiments in the following experimental setting: (1) the cardinality Card of input datasets varies in the range \([1 \times 10^5, 7 \times 10^5]\); (2) the parameter \( K \) is fixed to 3; (3) the dimensionality \( \zeta \) of full space is fixed to 8; (4) the number of subspaces varies in the range \([20, 60]\) (shown in Fig. 1, Fig. 2 and Fig. 3 show the results of experiments for independent datasets and anti-correlated datasets, respectively.

From Fig. 2 and Fig. 3, we can make the following observations:

(1) The AOMSR algorithm evidently outperforms the EARG solution in all cases. This is mainly because the AOMSR algorithm organizes all needed subspace L-EXTSKY recommendations as a subspace recommendation tree, and then utilizes the share mechanism in tree paths to reduce the total recommendation time. While the EARG solution simply runs the EARG approach \( u \) times where \( u \) is the number of subspaces. For example, in Fig. 2b, when the number of subspace is equal to 60, the recommendation time of the EARG solution exceeds 1951.3 seconds, while the AOMSR algorithm only needs 800.9 seconds. That is, in this case, the recommendation time of the AOMSR algorithm is only about 41.04 % of that of the EARG solution.
tion are about 5455,31 seconds and 2262,70 seconds, respectively. That is, in this case, the AOMSR algorithm can reduce the recommendation time by as much as 3192,61 seconds.

(3) The superiority of the AOMSR algorithm over the EARG solution is more marked for anti-correlated datasets than for independent ones. The main reason is that the EARG solution directly runs the EARG approach on original datasets and requires more recommendation time for anti-correlated datasets. For example, in Fig. 2a, when the number of subspace is equal to 50, the recommendation times of the AOMSR algorithm and the EARG solution are about 5,87 seconds and 1444,80 seconds, respectively. That is, in this case, the AOMSR algorithm only reduces the recommendation time by as much as 6,72 seconds. While in Fig. 3a, when the number of subspace is equal to 50, the recommendation times of the AOMSR algorithm and the EARG solution are about 1444,80 seconds and 4672,08 seconds, respectively. That is, in this case, the AOMSR algorithm can reduce the recommendation time by as much as 3227,28 seconds.

5.2 Evaluating two variations of the subspace L-extSKY recommendation

In this subsection, we focus on evaluating the efficiency of processing our two variations of subspace L-extSKY recommendation (i.e., global constraint L-extSKY recommendation and local constraint L-extSKY recommendation). The compared approaches are G_EARG (L_EARG) proposed in Section 4, and G_SUBSKY (L_SUBSKY) proposed in [7].

We implement this set of experiments in the following experimental setting: (1) the cardinality Card of every dataset varies in the range [1×10^5, 7×10^5]; (2) the parameter K is fixed to 3; (3) the dimensionality ζ of full space is fixed to 8; (4) the dimensionality v of subspace varies in the range [3, 7]; (5) for each dimension z, the constraints λ select the region [min_z+(max_z−min_z)/5, min_z+(max_z−min_z)/4/5] where min_z and max_z are the minimum and the maximum values of objects in every dataset on z, respectively. Fig. 4 and Fig. 5 show the results of experiments for independent datasets and anti-correlated datasets, respectively.

From Fig. 4 and Fig. 5, we can observe that the G_EARG and L_EARG respectively outperform the G_SUBSKY and L_SUBSKY in all cases. This is mainly because the EARG approach evidently outperforms the SUBSKY approach [7] in all cases. For example, in Fig. 4b, when the dimensionality of subspace is equal to 7, the recommendation time of the G_SUBSKY algorithm exceeds 1697,46 seconds, while the G_EARG algorithm only needs 298,17 seconds. That is, in this case, the recommendation time of the G_EARG algorithm is only about 17,57 % of that of the G_SUBSKY algorithm.

6 Conclusions and future work

In multi-user environments, the systems generally handle multiple subspace L-extSKY recommendations simultaneously. Hence, in this paper, we propose an efficient algorithm AOMSR to evidently reduce the total response time. The AOMSR algorithm first organizes all needed subspace L-extSKY recommendations as a subspace recommendation tree, and then employs the share mechanism in tree paths to enhance the total performance. Moreover, we discuss two interesting variations of subspace L-extSKY recommendation which are meaningful in practice, and show how our algorithm can be applied for their efficient processing. We also present the detailed predication condition which can cause the equivalence of these two variations. The detailed theoretical analyses and extensive experiments demonstrate that our proposed solution is both efficient and effective.

Future work will focus on using some more efficient index structures to improve the performance of our algorithm, extending our algorithms to stream and P2P environments, and on more experimentation.

Acknowledgements

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 Tung Education Foundation (No. 142002), the National Basic Research Program of China (973 Program 2014CB340404), and the Fundamental Research Funds for the Central Universities (Tongji University).

7 References

Efficient optimization for $L$-extSKY recommendations

Z. Huang et al.


[7] Tao, Y. F.; Xiao, X.; Pei, J. Efficient Skyline and Top-k Retrieval in Subspace. // IEEE Transactions on Knowledge and Data Engineering. 19, 8(2007), pp. 1072-1088. DOI: 10.1109/TKDE.2007.1051


Authors’ addresses

Zhenhua Huang, PhD.
Department of Computer Science, Tongji University
4800 Cao'an Hwy, Jiading
Shanghai 201804, China
E-mail: huangzhenhua@tongji.edu.cn

Juru Wang, PhD.
Department of Medical Electronics and Information Engineering,
Shanghai Medical Instrumentation College
Shanghai 200093, China
E-mail: smic.wangjr@163.com

Bo Zhang, PhD.
Department of Computer Science, Shanghai Normal University
Shanghai 200234, China
E-mail: zhangbo.snu@gmail.com