ISSN 1330-3651 (Print), ISSN 1848-6339 (Online)
DOI: 10.17559/TV-20150731103906

EFFICIENT OPTIMIZATION FOR *L*-EXTSKY RECOMMENDATIONS

Zhenhua Huang, Juru Wang, Bo Zhang

Original scientific paper
L-extSKY 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*-extSKY objects in one single subspace. However, in multi-user environments, the system generally handles multiple subspace *L*-extSKY recommendations simultaneously. Hence, in this paper, we present an efficient algorithm AOMSR (Algorithm for Optimizing Multiple Subspace *L*-extSKY Recommendations) to remarkably reduce the total response time. Furthermore, we discuss two interesting variations of *L*-extSKY recommendation, i.e., global constraint *L*-extSKY recommendation and local constraint *L*-extSKY 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-extSKY recommendation; subspace; performance evaluation*

Učinkovita optimizacija za *L*-extSKY preporuke

Izvorni znanstveni članak
L-extSKY preporuka je nedavno privukla veliku pažnju pretraživatelja informacija. U literaturi [1] predlaže se algoritam EARG (Efficient Approach based on Regular Grid) za dobivanje *L*-extSKY objekata u jednom jedinom podprostoru. Međutim, u okruženju s više korisnika, sustav obično simultano rješava mnogostruke podprostorne *L*-extSKY preporuke. U ovom radu stoga predstavljamo učinkoviti algoritam AOMSR (Algorithm for Optimizing Multiple Subspace *L*-extSKY Recommendations) u svrhu značajnog smanjenja ukupnog vremena odziva. Nadalje, raspravljamo o dvije interesantne varijacije *L*-extSKY preporuke, tj. globalnom ograničenju *L*-extSKY preporuke i lokalnom ograničenju *L*-extSKY preporuke, koje su od praktičnog značaja i pokazuju kako se naš algoritam može primijeniti u svrhu njihove učinkovite obrade. Detaljna teoretska analiza i velik broj eksperimenata kojima se demonstrira naše rješenje su i efikasni i efektivni.

Ključne riječi: *pretraživanje informacija; L-extSKY preporuka; podprostor; ocjena dobivenih rezultata*

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 invariable.

Literature [1] points out that in most real applications, for a ζ-dimensional dataset AD, the cardinality of its recommendation result does not exceed $(\ln^{\zeta-1}|AD|)/(|AD|\cdot(\zeta-1)!)$ 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*-extSKY recommendation. Given a set of ζ-dimensional objects, a *L*-extSKY 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*-extSKY recommendation has at least 2 advantages: (1) the *L*-extSKY 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*-extSKY recommendation is more meaningful in practice. Furthermore, the literature [1] presents an algorithm EARG (Efficient Approach based on Regular Grid) to produce the *L*-extSKY 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*-extSKY recommendations simultaneously. Hence, in this paper, we propose an efficient algorithm AOMSR (Algorithm for Optimizing Multiple Subspace *L*-extSKY Recommendations) to markedly reduce the total recommendation time. The AOMSR algorithm first organizes all issued subspace *L*-extSKY recommendations as a recommendation tree, and then uses the share mechanism in tree paths to improve the total performance of these *L*-extSKY recommendations. Moreover, we discuss two interesting variations of *L*-extSKY recommendation, i.e., global constraint *L*-extSKY recommendation and local constraint *L*-extSKY 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 ζ 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_i ($1\le i\le\zeta$) be δ_i. Then the cell $c[a_1,\ldots, a_\zeta]$ contains all objects with the i-th dimension in the range $[a_i\cdot(\delta_i\text{-}1), a_i\cdot\delta_i)$. 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 $b_i(1\le i\le\zeta)=\lfloor p.x_i/\delta_i\rfloor$. 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 recommendation.

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

In order to efficiently realize the EARG approach, the literature [1] distinguishes three possibilities about the relationship between any two V-dimensional cells $C_1(V)=[a_1,\ldots, a_v]$ and $C_2(V)=[b_1,\ldots, b_v]$.

Definition 2. If two V-dimensional cells $C_1(V)=[a_1,\ldots, a_v]$ and $C_2(V)=[b_1,\ldots, b_v]$ satisfy the following condition, then we say "$C_1(V)$ fully dominates $C_2(V)$": $\forall i\in[1, v]$, $a_i<b_i$.

For simplicity, literature [1] denotes this relationship as $C_1(V)\lhd\lhd_V C_2(V)$.

Definition 3. If two V-dimensional cells $C_1(V)=[a_1,\ldots, a_v]$ and $C_2(V)=[b_1,\ldots, b_v]$ satisfy the following condition, then we say "$C_1(V)$ partially dominates $C_2(V)$": $\exists U\subset V$, $\forall i\in[1, |U|]$, $a_i=b_i$, and $\forall j\in[|U|, v]$, $a_i<b_i$.

For simplicity, literature [1] denotes this relationship as $C_1(V)\lhd_V C_2(V)$.

Definition 4. If two V-dimensional cells $C_1(V)=[a_1,\ldots, a_v]$ and $C_2(V)=[b_1,\ldots, b_v]$ satisfy the following condition, then we say "$C_1(V)$ is incomparable with $C_2(V)$": $\exists i\in[1, v]$, $a_i<b_i$, and $\exists j\in[1, v]$, $b_j<a_j$.

For simplicity, literature [1] denotes this relationship as $C_1(V)\rhd\lhd_V C_2(V)$.

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

(a) $C_1(V)\lhd\lhd_V C_2(V)\Rightarrow\forall p\in S_1(V)$, $\forall r\in S_1(V)$, r is dominated by p;

(b) $C_1(V)\lhd_V C_2(V)\Rightarrow\forall p\in S_1(V)$, $\forall r\in S_1(V)$, p is not dominated by r;

(c) $C_1(V)\rhd\lhd_V C_2(V)\Rightarrow\forall p\in S_1(V)$, $\forall r\in S_1(V)$, p (r) is not dominated by r (p).

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 ζ-dimensional objects AD, each object is associated with one counter *count* whose initial value equals 0; the subspace V and its dimensionality v; the regular grid index $\Xi(AD, F)$; the parameter L.
Output: the L-extSKY set $L\text{-}EXSET(AD, V)$.
Begin

1. $L\text{-}EXSET(AD, V)\leftarrow\varnothing$;
2. Obtain all V-dimensional cells which contain at least one object from $\Xi(AD, F)$;
3. Organize these non-empty cells as a sequence $seqV$ which satisfies: for any two cells $C_\alpha(V)=[a_1,\ldots, a_v]$ and $C_\beta(V)=[b_1,\ldots, b_v]$, if $\alpha<\beta$, then $\Sigma_{i=1}^{v}a_i\le\Sigma_{i=1}^{v}b_i$;
4. For orderly visit each cell $C_\alpha(V)$ in $seqV$ Do
5. $S_\alpha(V)\leftarrow$the set of objects inside $C_\alpha(V)$;
6. $CR(C_\alpha(V))\leftarrow$**OBCANS**$(S_\alpha(V), V, L)$;
   /* the function **OBCANS**$(S_\alpha(V), V, L)$ returns all candidate L-extSKY objects in $S_\alpha(V)$, which is shown in Algorithm 2 */
7. $flag\leftarrow$False;
8. For orderly visit each cell $C_\lambda(V)$ in $seqV$ which Locates before $C_\alpha(V)$ Do
9. If $C_\alpha(V)\lhd\lhd_V C_\lambda(V)$Then
10. For $\forall p\in CR(C_\alpha(V))$ Do
11. $p.count\leftarrow p.count+|S_\lambda(V)|$;
12. If $p.count\ge L$ Then $flag\leftarrow$True;
13. $CR(C_\alpha(V))\leftarrow CR(C_\alpha(V))\text{-}\{p|\ p.count>L\}$;
14. If $C_\alpha(V)\lhd_V C_\lambda(V)$Then
15. For $\forall p\in CR(C_\alpha(V))$, $\forall r\in S_\lambda(V)$ Do
16. If p dominates r Then
17. $p.count\leftarrow p.count+1$;
18. If $p.count\ge L$ Then $flag\leftarrow$True;
19. $CR(C_\alpha(V))\leftarrow CR(C_\alpha(V))\text{-}\{p|\ p.count>L\}$;
20. If $flag$=True Then $seqV\leftarrow seqV\text{-}\{C_z(V)|\ C_z(V)\lhd\lhd_V C_\alpha(V)\}$;
21. $L\text{-}EXSET(AD, V)\leftarrow L\text{-}EXSET(AD, V)\cup CR(C_\alpha(V))$;
22. Return $L\text{-}EXSET(AD, V)$;

End

In the step 3 of Algorithm 1, any two cells $C_\alpha(V)=[a_1,\ldots, a_v]$ and $C_\beta(V)=[b_1,\ldots, b_v]$ in $seqV$ need to satisfy the condition: $\alpha<\beta\Rightarrow\Sigma_{i=1}^{v}a_i\le\Sigma_{i=1}^{v}b_i$, which has at least two advantages: (1) If a cell $C_\alpha(V)$ is visited earlier, then the probability that it is fully or partially dominated by other cells is lower. (2) If a cell $C_\alpha(V)$ locates before another cell $C_\beta(V)$, then $C_\beta(V)$ cannot fully or partially dominate $C_\alpha(V)$, and hence according to Theorem 1, we can have that the objects in $S_\beta(V)$ cannot dominate any object in $S_\alpha(V)$ on V. Consequently, this condition can remarkably reduce the number of comparisons between objects. For each visited cell $C_\alpha(V)$, the step 6 uses the function OBCANS which is described in Algorithm 2 to obtain all candidate L-extSKY objects in $S_\alpha(V)$; that is, if $p\in CR(C_\alpha(V))$, then p is dominated by at most K objects in $S_\alpha(V)$ on V. In the steps 8÷19, for each object $p\in CR(C_\alpha(V))$, the algorithm modifies its counter. And if $p.count$ exceeds L then p can be safely removed from $CR(C_\alpha(V)$. Specially, the steps 9÷13 focus on considering each cell $C_\lambda(V)$ which fully dominates $C_\alpha(V)$, while the steps 14÷19 focus on considering each cell $C_u(V)$ which partially dominates $C_\alpha(V)$. In the steps 9÷13, if $C_\lambda(V)$ fully dominates $C_\alpha(V)$, 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 $C_\lambda(V)$, and *p.count* is directly added $|S_\lambda(V)|$. While in the steps 14÷19, if $C_u(V)$ 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 $S_u(V)$, and *p.count* is added the number of objects in $S_u(V)$ 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-extSKY objects in $S_\alpha(V)$;

Begin

1. $CR(C_\alpha(V)) \leftarrow \varnothing$;
2. For each object $p \in S_\alpha(V)$ Do $p.key \leftarrow \Sigma_{i=1}^{v} \ln(p[i]+1)$;
3. $L_\alpha(V) \leftarrow$the object list obtained by sorting all the objects in $S_\alpha(V)$ in the key ascending order;
4. $flag_1 \leftarrow$False;
5. For orderly visit each object $p \in L_\alpha(V)$ Do
6. $flag_2 \leftarrow$False;
7. For orderly visit each object $r \in L_\alpha(V)$ which locates before p Do
8. If p dominates r Then
9. $p.count \leftarrow p.count+1$;
10. If $p.count \geq L$ Then $flag_1 \leftarrow$True;
11. If $p.count > L$ Then $flag_2 \leftarrow$True; break;
12. If $flag_2$=False Then $CR(C_\alpha(V)) \leftarrow CR(C_\alpha(V)) \cup \{p\}$;
13. If $flag_1$=True Then
14. $seqV \leftarrow seqV - \{C_z(V) | C_z(V) \lhd\lhd_V C_\alpha(V)\}$;
15. Return $CR(C_\alpha(V))$;

End

In the steps 2÷3 of Algorithm 2, we sort the objects in $S_\alpha(V)$ using the sorting-operator $\Sigma_{i=1}^{v} \ln(p[i]+1)$, 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 $\Sigma_{i=1}^{v} \ln(p[i]+1)$ 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.count*. And if *p.count* does not exceed L, then the algorithm adds p into $CR(C_\alpha(V))$. Furthermore, in the algorithm, we use the Boolean variant $flag_1$ as a cell-pruning indicator. When $flag_1$ is 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*-extSKY recommendations simultaneously

This section focuses on optimizing multiple subspace L-extSKY recommendations simultaneously in multi-user environments. Let u be the number of subspace L-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-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 $p.count_V$ 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) $\forall V_i, V_j \in SQB$, $i<j \Rightarrow |V_i| \geq |V_j|$, where $|V_i|$ is the dimensionality of V_i.

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).

(*a*) V_0 is the root of T_{sb};

(*b*) $ND=\{V_i | V_i \in SQB\}$.

Property 2 (of $<V_i, V_j> \in ES$*).*

(*a*) $V_j \subset V_i$;

(*b*) $\neg\exists V_g \in ND$, $V_i \supset V_g \wedge V_g \supset V_j$;

(*c*) $\neg\exists V_h \in ND$, such that $<V_i, V_j> \in ES$ and $<V_i, V_j>$ satisfies Property 2 (*a*) and (*b*), and $h<i$.

Definition 7. Let AD be the set of ζ-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-extSKY set for each node in T_{sb} as follows:

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

Definition 8. Let AD be the set of ζ-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-extSKY set, we can define the shadow subspace L-extSKY set for each node in T_{sb} as follows:

$$rep(V) = \begin{cases} \varnothing; \quad iff\ V = rt. \\ \{p \mid p \in AD - seed(PV) \wedge \exists r \in seed(V), \\ \quad \forall z \in V, p[z] = r[z]; \quad otherwise. \end{cases}$$

Theorem 2. Let AD be the set of ζ-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_{sb}, we can have:

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

Proof: Assume the root node rt is at level 1 in the tree T_{sb}. Then, for rt, we can have: $L\text{-}EXSET(seed(rt)\cup rep(rt), rt)=L\text{-}EXSET(L\text{-}EXSET(AD, rt)\cup\varnothing, rt)=L\text{-}EXSET(AD, rt)$. So, the theorem holds for the root node rt. Below, we focus on each non-root node V and prove that: (1) $L\text{-}EXSET(AD, V)\subseteq L\text{-}EXSET(seed(V)\cup rep(V), V)$, and (2) $L\text{-}EXSET(seed(V)\cup rep(V), V)\subseteq L\text{-}EXSET(AD, V)$. For (1), we prove its correctness by contradiction. Assume there exists an object p such that $p\in L\text{-}EXSET(AD, V)$ and $p\notin L\text{-}EXSET(seed(V)\cup rep(V), V)$. Since $p\notin L\text{-}EXSET(seed(V)\cup rep(V), V)$, there exists a set $\Im$ such that $|\Im|>L$ and $\forall r\in\Im$, p dominates r. And since $seed(V)\cup rep(V)\subseteq AD$, $p\notin L\text{-}EXSET(AD, V)$. This contradicts the assumption that $p\in L\text{-}EXSET(AD, V)$. Therefore, $L\text{-}EXSET(AD, V)\subseteq L\text{-}EXSET(seed(V)\cup rep(V), V)$. For (2), we also prove its correctness by contradiction. Assume there exists an object p such that $p\in L\text{-}EXSET(seed(V)\cup rep(V), V)$ and $p\notin L\text{-}EXSET(AD, V)$. From the assumption, we can easily see that there exists some object q such that $q\in AD\text{-}(seed(V)\cup rep(V))$ and p dominates q. Since $q\notin seed(V)\cup rep(V)$, $q\notin L\text{-}EXSET(seed(V)\cup rep(V), V)$. Hence, $q.count_V>L$. Therefore, we can have: $p.count_V>L$. That is, $p\notin L\text{-}EXSET(seed(V)\cup rep(V), V)$. This contradicts the above assumption that $p\in L\text{-}EXSET(seed(V)\cup rep(V), V)$. So then, $L\text{-}EXSET(seed(V)\cup rep(V), V)\subseteq L\text{-}EXSET(AD, V)$. 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 ζ-dimensional objects AD; the regular grid index $\Xi(AD, F)$; u L-extSKY recommendations whose corresponding subspaces are $V_1,\ldots, V_u$.

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

Begin

1. $SQB\leftarrow$the consistent subspace sequence containing u subspaces;
2. $T_{sb}\leftarrow$the subspace recommendation tree over SQB;
3. For each node V in T_{sb} Do
4. If V is the root node rt Then
5. $L\text{-}EXSET(AD, rt)\leftarrow$EARG($AD$, rt, $\Xi(AD, F)$, L);
6. Else
7. $PV\leftarrow$the parent node of V;
8. $seed(V)\leftarrow$EARG($seed(PV)$, V, $\Xi(AD, F)$, L);
9. $\Delta(V)\leftarrow seed(V)$;
10. Divide $seed(V)$ into m subsets $SD_1,\ldots, SD_m$ which satisfy the following conditions: (a) $\forall p, r\in SD_i, \forall z\in V, p[z]=r[z]$, and ($b$) $\forall p\in SD_i, r\in SD_j, i\neq j\Rightarrow\exists\, z\in V, p[z]\neq r[z]$;
11. For each SD_i, $i\in[1, m]$ Do
12. Select some object p from SD_i;
13. $SD_i.key\leftarrow\Sigma_{z=1}^{v} p[z]$;
14. Organize these m subsets as a list $LD=<SD_1,\ldots, SD_m>$ which satisfies: $i<t\Rightarrow SD_i.key\leq SD_t.key$;
15. For each subset SD_i in LD Do
16. Select some object p from SD_i;
17. $SB_i\leftarrow\{q|\ q\in AD\text{-}\Delta(PV)\wedge q$ falls inside the covering cell of $p\wedge\forall z\in[1, v], q[z]=p[z]\}$;
18. For each object $q\in SB_i$ Do $q.count_V\leftarrow p.count_V$;
19. For each SD_t which locates after SD_i in LD Do
20. Select some object r from SD_t;
21. If r dominates p Then
22. If $r.count_V+|SB_i|>L$ Then
23. $seed(V)\leftarrow seed(V)\text{-}SD_t$;
24. $LD\leftarrow LD\text{-}\{SD_t\}$;
25. Else For each object $\delta\in SD_t$ Do
26. $\delta.count_V\leftarrow\delta.count_V+|SB_i|$;
27. $rep(V)\leftarrow rep(V)\cup SB_i$;
28. $L\text{-}EXSET(AD, V)\leftarrow seed(V)\cup rep(V)$;
29. Return u subspace $L\text{-}extSKY$ 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_{sb} (Steps 1÷2), the AOMSR algorithm visits each node V and obtains its subspace L-extSKY 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-extSKY set is directly obtained from the original dataset AD using the EARG approach, $seed(rt)=L\text{-}EXSET(AD, rt)$. 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-extSKY set $seed(V)$ in step 8, and then uses $seed(V)$ to produce the correct subspace L-extSKY set $L\text{-}EXSET(AD, V)$ in the succeeding steps. In the step 10, the algorithm divides $seed(V)$ 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 $SD_i.key$. Then the step 14 organizes these m subsets as a list $LD=<SD_1,\ldots, SD_m>$ satisfying: $i<t\Rightarrow SD_i.key\leq SD_t.key$. Note that for any two subsets SD_i and SD_t in LD, if SD_i locates before SD_t, then the objects in SD_t 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 $AD\text{-}\Delta(PV)$ 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 $q.count_V$ to be equal to $p.count_V$. Since the objects in SB_i are added into the shadow subspace L-extSKY set $rep(V)$ and ultimately are incorporated into $L\text{-}EXSET(AD, V)$ (steps 27÷28), the algorithm needs to handle each subset SD_t which locates after SD_i in LD (steps 19÷26). The handling process can be described as follows. The algorithm first randomly selects an object r from SD_t and checks if r is dominated by p on V. If yes, then the algorithm continues to evaluate the value of $r.count_V+|SB_i|$. If $r.count_V+|SB_i|$ is greater than K, then the algorithm deletes all the objects in SD_t from $seed(V)$ and removes SD_t from LD; otherwise, for each object δ in SD_t, $\delta.count_V$ is added $|SB_i|$.

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) , p_{10}(7.0, 4.6)}. 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_{sb}: $V_1 \rightarrow V_2$, 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-$EXSET(AD, V_1)$=$seed(V_1)$= {p_1, p_2, p_4, p_8, p_5, p_6}. 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_5, p_6}. Then the AOMSR algorithm divides $seed(V_2)$ into two subsets SD_1={p_5} and SD_2={p_6}, and organizes these two subsets as a list LD=<SD_2, SD_1>. The AOMSR algorithm orderly handles these two subsets. For SD_2, the AOMSR algorithm first randomly selects an object (i.e., p_6) from SD_2, and then obtains the set $SB_2=\varnothing$. Since SB_2 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_5) from SD_1, and then obtains the set SB_1={p_9}. Then, the AOMSR algorithm sets $p_9.count_{V2}$ to be equal to $p_5.count_{V2}$ (i.e., 1) and adds p_9 into $rep(V_2)$. Hence, l-$EXSET(AD, V_2)$=$seed(V_1) \cup rep(V_2)$={p_5, p_6, p_9}.

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-$EXSET(AD, V)$ 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 v-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 ζ-dimensional objects, and V={d_1,…, d_2} be the issued subspace. And for each $d_i \in V$, users are only interested in the region λ_i=[B_i, E_i]. Then the global constraint L-extSKY recommendation on V returns the set L-$GEXSET(AD, V, \lambda)$={$p | p \in L$-$EXSET(AD, V) \wedge \forall i \in [1, v]$, $B_i \le p[i] \le E_i$}, where $\lambda = \lambda_1 \wedge \ldots \wedge \lambda_v$.

From Definition 9, we can easily see that L-$GEXSET(AD, V, \lambda)$ is the subset of L-$EXSET(AD, V)$ and consists of the objects which belong to L-$EXSET(V, AD)$ and fall inside the data space defined by the constraints λ. For simplicity, we call L-$GEXSET(AD, V, \lambda)$ *global constraint subspace L-extSKY set*. Furthermore, if $\forall i \in [1, v]$, $B_i \le p[i] \le E_i$, then we say "p satisfies the constraints λ" which is denoted as $\lambda(p)$.

Example 2. Returning to Example 1, 1-$EXSET(AD, V_1)$= {p_1, p_2, p_4, p_8, p_5, p_6}. When the user Jukie gives the constraints $\lambda = \lambda_1 \wedge \lambda_2$ where λ_1=[2, 4] and λ_2=[3, 4]. Then, 1-$GEXSET(AD, V_1, \lambda)$={p_2, p_4, p_8}.

Definition 10. Let AD be the set of ζ-dimensional objects, and V={d_1,…, d_2} be the issued subspace. And for each $d_i \in V$, users are only interested in the region λ_i=[B_i, E_i]. Then the local constraint L-extSKY recommendation on V returns the set L-$CEXSET(AD, V, \lambda)$=L-$EXSET(SD, V)$, where SD={$p | p \in AD \wedge \forall i \in [1, v]$, $B_i \le p[i] \le E_i$} and $\lambda = \lambda_1 \wedge \ldots \wedge \lambda_v$.

From Definition 10, we can see that L-$CEXSET(AD, V, \lambda)$ 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 λ. It is not difficult to see that L-$CEXSET(AD, V, \lambda) \subset L$-$EXSET(SD, V)$. For simplicity, we call L-$CEXSET(AD, V, \lambda)$ *local constraint subspace l-SkyRex set*.

Example 3. Returning to Example 1, 1-$EXSET(AD, V_1)$= {p_1, p_2, p_4, p_8, p_5, p_6}. When the user Jukie gives the constraints $\lambda = \lambda_1 \wedge \lambda_2$ where λ_1=[8, 10] and λ_2=[2, 3]. Then, 1-$CEXSET(AD, V_1, \lambda)$={p_3, p_9}.

It is important to note that we can easily obtain the global constraint subspace L-extSKY set and the local constraint subspace *L-extSKY* set by slightly modifying the EARG approach. For the former, we only need to modify the step 21 of EARG: L-$EXSET(AD, V) \leftarrow L$-$EXSET(AD, V) \cup \{r | r \in CR(C_\alpha(V) \wedge \forall i \in [1, v], B_i \le r[i] \le E_i)$. While for the latter, we only need to replace the set AD as the set SD={$p | p \in AD \wedge \forall i \in [1, v]$, $B_i \le p[i] \le E_i$}. Furthermore, generally given the constraints λ, L-$GEXSET(AD, V, \lambda) \neq L$-$CEXSET(AD, V, \lambda)$. However, the following theorem shows that if the constraints λ satisfy the condition $\forall p$, $r \in AD$, $\lambda(p) \wedge p$ dominates $r \Rightarrow \lambda(r)$, then L-$GEXSET(AD, V, \lambda) \neq L$-$CEXSET(AD, V, \lambda)$.

Theorem 3. *L*et AD be the set of ζ-dimensional objects, and V be the issued subspace. And users are only interested in the data space defined by the constraints λ. Then we can have: $\forall p$, $r \in AD$, $\lambda(p) \wedge p$ dominates $r \Rightarrow \lambda(r) \Leftrightarrow L$-$GEXSET(AD, V, \lambda) \neq L$-$CEXSET(AD, V, \lambda)$.

Proof: For every z ($0 \le z \le L$), we let L-$EXSET(AD, V)^{(z)}$ be the set of objects in L-$EXSET(AD, V)$ whose values of counter equal z, L-$GEXSET(AD, V, \lambda)^{(z)}$ be the set of objects in L-$GEXSET(AD, V, \lambda)$ whose values of counter equal z, and L-$CEXSET(AD, V, \lambda)^{(z)}$ be the set of objects in L-$CEXSET(AD, V, \lambda)$ whose values of counter equal z. Obviously,

$$L\text{-}EXSET(AD, V, \lambda) = \bigcup_{z=0}^{L} L - EXSET(AD, V, \lambda)^{(z)},$$

$$L\text{-}GEXSET(AD, V, \lambda) = \bigcup_{z=0}^{L} L - GEXSET(AD, V, \lambda)^{(z)},$$

$$L\text{-}CEXSET(AD, V, \lambda) = \bigcup_{z=0}^{L} L - CEXSET(AD, V, \lambda)^{(z)}.$$

In the following part, we will prove by induction on the z ($0 \le z \le L$) that $\forall p$, $r \in AD$, $\lambda(p) \wedge (p$ dominates $r) \Rightarrow \lambda(r) \Leftrightarrow L$-$GEXSET(AD, V, \lambda)^{(z)} = L$-$CEXSET(AD, V, \lambda)^{(z)}$.

(1) *Base Case*. For z=0, it is easy to see that L-$EXSET(AD, V)^{(0)}$ is the skyline set of AD on V. On the other hand, we can have: L-$GEXSET(AD, V, \lambda)^{(0)} = \lambda(L$-$EXSET(AD, V)^{(0)})$, and L-$CEXSET(AD, V, \lambda)^{(0)} = L$-$EXSET(\lambda(AD), V)^{(0)}$. (*i*) "$\Rightarrow$": we are necessary to prove that if $\forall p$, $r \in AD$, $\lambda(p) \wedge (p$ dominates $r)$, then $\lambda(L$-$EXSET(AD, V)^{(0)}) = L$-$EXSET(\lambda(AD), V)^{(0)}$. Since, $\forall p$, $r \in AD$, $\lambda(p) \wedge (p$ dominates $r) \Rightarrow \lambda(r)$, for each $p \in AD$, we only need to focus on the objects which dominate p on V and fall inside the data space defined by λ. And hence, $\lambda(L$-$EXSET(AD, V)^{(0)}) = L$-$EXSET(\lambda(AD), V)^{(0)}$. (*ii*) "$\Leftarrow$": we prove its correctness by contradiction. Assume that when $\lambda(L$-$EXSET(AD, V)^{(0)}) = L$-$EXSET(\lambda(AD), V)^{(0)}$, $\forall p$, $r \in AD$, $\lambda(p) \wedge (p$ dominates $r) \not\Rightarrow \lambda(r)$. Since $\lambda(p) \wedge (p$ domi-

nates $r) \Rightarrow \lambda(r) \Leftrightarrow \neg(\lambda(p) \wedge (p$ dominates $r)) \vee \lambda(r)$; $\exists\ p$, $r \in AD$, $\lambda(p) \wedge (p$ dominates $r) \wedge \sim\lambda(r)$. Hence, $\exists\ p$, $p \in L\text{-}EXSET(\lambda(AD), V)^{(0)} \Rightarrow \exists\ p, p \notin \lambda(L\text{-}EXSET(AD, V)^{(0)})$. This contradicts with the above assumption that $\lambda(L\text{-}EXSET(AD, V)^{(0)}) = L\text{-}EXSET(\lambda(AD), V)^{(0)}$. Hence, the theorem stands.

(2) *Inductive Hypothesis*. For $z \le n$, $\forall p, r \in AD$, $\lambda(p) \wedge (p$ dominates $r) \Rightarrow \lambda(r) \Leftrightarrow \lambda(L\text{-}EXSET(AD, V)^{(n)}) = L\text{-}EXSET(\lambda(AD), V)^{(n)}$.

(3) *Inductive Case*. For $z = n+1$, we will prove the correctness of the following formula: $\forall p, r \in AD$, $\lambda(p) \wedge (p$ dominates $r) \Rightarrow \lambda(r) \Leftrightarrow \lambda(L\text{-}EXSET(AD, V)^{(n+1)}) = L\text{-}EXSET(\lambda(AD), V)^{(n+1)}$. According to the hypothesis: for $z \le n$, $\forall p, r \in AD$, $\lambda(p) \wedge (p$ dominates $r) \Rightarrow \lambda(r) \Leftrightarrow \lambda(L\text{-}EXSET(AD, V)^{(n)}) = L\text{-}EXSET(\lambda(AD), V)^{(n)}$, we have $\forall p, r \in AD$, $\lambda(p) \wedge (p$ dominates $r) \Rightarrow \lambda(r) \Leftrightarrow \bigcup_{i=1}^{n} \lambda(L\text{-}EXSET(AD, V)^{(i)}) = \bigcup_{i=1}^{n} L\text{-}EXSET(\lambda(AD), V)^{(i)}$. Hence, $\lambda(p) \wedge (p$ dominates $r) \Rightarrow \lambda(r) \Leftrightarrow \lambda(L\text{-}EXSET(AD, V)^{(0)} - \bigcup_{i=0}^{n} L\text{-}EXSET(AD, V)^{(i)}) = L\text{-}EXSET(\lambda(AD), V)^{(0)} - \bigcup_{i=1}^{n} L\text{-}EXSET(\lambda(AD), V)^{(i)}$. That is, $\lambda(p) \wedge (p$ dominates $r) \Rightarrow \lambda(r) \Leftrightarrow \lambda(L\text{-}EXSET(AD, V)^{(n+1)}) = L\text{-}EXSET(\lambda(AD), V)^{(n+1)}$.

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. And 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 $[1\times10^5, 7\times10^5]$; (2) the parameter *K* is fixed to 3; (3) the dimensionality ζ 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.

the dimensionality of subspace	the number of subspaces				
	20	30	40	50	60
7	1	1	1	1	1
6	2	3	4	5	6
5	4	6	8	10	12
4	8	12	16	20	24
3	5	8	11	14	17

Figure 1 The relationship between the number of subspaces and the dimensionality of subspace

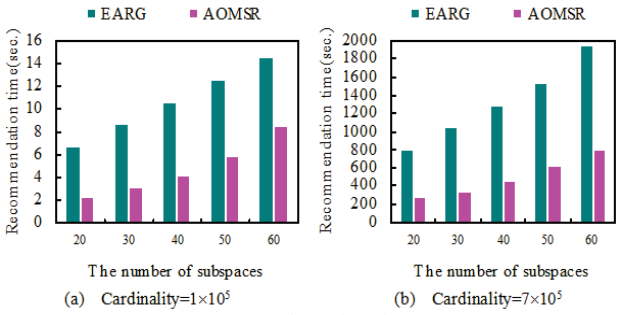

Figure 2 Independent datasets

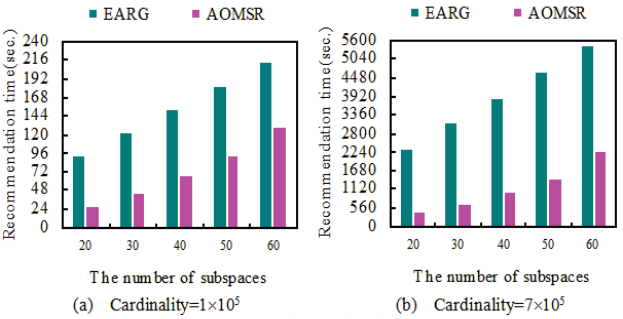

Figure 3 Anti-correlated datasets

(2) The superiority of the AOMSR algorithm over the EARG solution becomes more marked as the number of subspaces increases. This is mainly because when the number of subspaces increases, the number of *L*-extSKY recommendations which need not to be answered from the original datasets will increase, and hence the sharing effect of the AOMSR algorithm is more evident. For example, in Fig. 3b, when the number of subspaces is equal to 20, the recommendation times of the AOMSR algorithm and the EARG solution are about 469,60 seconds and 2323,29 seconds, respectively. That is, in this case, the AOMSR algorithm only reduces the recommendation time by as much as 1853,69 seconds. While when the number of subspaces is equal to 60, the recommendation times of the AOMSR algorithm and the EARG solu-

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 12,59 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 $[1\times10^5, 7\times10^5]$; (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 $[min_z+(max_z-min_z)/5, min_z+(max_z-min_z)\times4/5]$ 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.

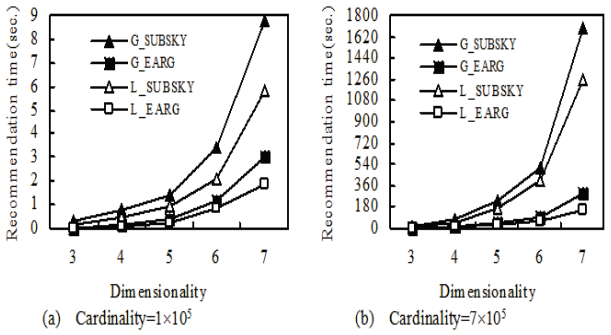

Figure 4 Independent datasets

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.

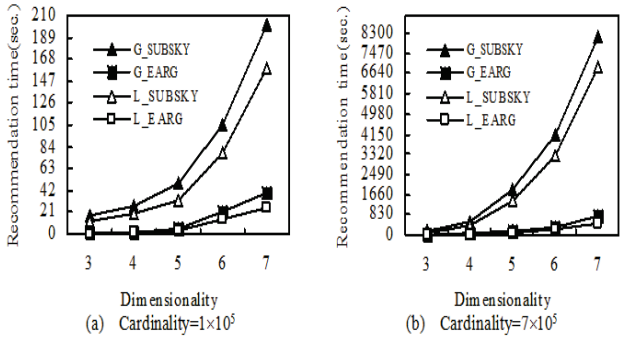

Figure 5 Anti-correlated datasets

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

[1] Lin, Z. K.; Huang, Z. H.; Xiang, Y. Efficient Processing of k-Quasi Skyline Query. // Systems Engineering -Theory & Practice. 32, 5(2012), pp. 1098-1106.

[2] Son, W. B.; Hwang, S. W.; Ahn, H. K. MSSQ: Manhattan Spatial Skyline Queries. // Information Systems. 40, 3(2014), pp. 67-83. DOI: 10.1016/j.is.2013.10.001

[3] Rajasekaran, S.; Zhang, N. On Skyline Groups. // IEEE Transactions on Knowledge and Data Engineering. 26, 4(2014), pp. 942-956. DOI: 10.1109/TKDE.2013.119
[4] Chomicki, J.; Godfrey, P.; Gryz, J.; Liang, D. Skyline with Presorting: Theory and Optimization. // Proceedings of the Int. conference on Intelligent Information Systems / Gdansk, 2005, pp. 595-604. DOI: 10.1007/3-540-32392-9_72
[5] Xia, T.; Zhang, D.; Fang, Z.; Chen, C.; Wang, J. Online Subspace Skyline Query Processing Using the Compressed Skycube. // ACM Transactions on Database Systems. 37, 2(2012), pp. 1-36. DOI: 10.1145/2188349.2188357
[6] Tambaram, K. G.; Lee, J. S.; Rhee, J. W.; Rhee, J. W.; Kang, J. Efficient Skycube Computation Using Point and Domain-based Filtering. // Information Sciences. 180, 7(2010), pp. 1090-1103. DOI: 10.1016/j.ins.2009.11.040
[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
[8] Lee, J.; Hwang, S. Toward Efficient Multidimensional Subspace Skyline Computation. // The VLDB Journal, 23, 1(2014), pp. 129-145. DOI: 10.1007/s00778-013-0317-y
[9] Godfrey, P. Skyline Cardinality for Relational Processing. // Proceedings of the Int. conference on Foundations of Information and Knowledge Systems / Vienna, 2004, pp. 78-97. DOI: 10.1007/978-3-540-24627-5_7
[10] Huang, Z.; Xiang, Y.; Zhang, B.; Wang, D.; Liu, X. L. An Efficient Method for K-Means Clustering. // Pattern Recognition and Artificial Intelligence, 23, 4(2010), pp. 516-521.

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