数理解析研究所講究録
第 1599 巻 2008 年 162-169

Notes on Enumeration of Concepts in a Sperner Family Concept Class Using Subconcept Queries

Atsuyoshi Nakamura and **Mineichi Kudo**
Graduate School of Information Science and Technology, Hokkaido University
{atsu,mine}@ main.ist.hokudai.ac.jp

Abstract

In this paper, we study the problem of enumerating *concepts* in a *Sperner family concept class* using *subconcept queries*, which was first studied by Gunopulos et al. [3]. We characterize the complexity of Sperner family concept classes by two measures, *VC dimension* and *characteristic dimension*. We show algorithms which can efficiently enumerate concepts in a SpernerFamily concept class with small VC dimension, including a new algorithm related to the introduced two measures.

1 Introduction

Enumeration is very popular in the area of data mining. Many enumeration problems for itemsets, sequences, trees and graphs that satisfy the constraint of frequency have been studied so far. Among them, problems of enumeration of maximal patterns only [4, 6] are very important to reduce the complexity of computational time and outputted results.

In this paper, we study a general problem, the problem of enumerating *concepts* in a *Sperner family concept class* using *subconcept queries*, which was first studied by Gunopulos et al. [3]. Here, a *concept* is a subset of a fixed finite set and a *Sperner family concept class* is a family of concepts in which no concept is a superset of any other concept. A *subconcept query* for subset T is a query that asks if there is a concept which contains T. In the case of enumerating maximal frequent itemsets, a maximal frequent itemset C is a concept, and a query that asks if an itemset is frequent is a subconcept query. In addition to enumeration of maximal frequent itemsets, this problem is known to have many instance applications such as minimal keys in a relational database and learning monotone boolean functions [3].

In this paper, we characterize the complexity of Sperner family concept classes $\mathcal{C}$ by two measures, the VC dimension $d_{\mathcal{C}}$ of intersection closure of $\mathcal{C}$ and its *characteristic dimension*. The *characteristic dimension of a concept* C is defined as the minimum number of elements in C needed to identify it uniquely, and the *characteristic dimension of a concept class* $\mathcal{C}$ is the maximum characteristic dimension of any $C \in \mathcal{C}$. As for relation to hypergraph theory, $d_{\mathcal{C}} + 1$ is shown to be an upper bound of the *rank* of the transversal hypergraph $\overline{\mathcal{C}}$, where $\overline{\mathcal{C}}$ is the set of the compliment sets of Cs in $\mathcal{C}$.

When both $|\mathcal{C}|$ and $d_{\mathcal{C}}$ are small, concepts in $\mathcal{C}$ can be enumerated efficiently. As one of such efficient algorithms, we propose algorithm EnumSpernerFamily, which is related to the two measures introduced above. EnumSpernerFamily is an algorithm of repeatedly executing the procedure that finds a seed (subset) of a concept which has not found yet and obtains one concept by making the seed grow up. Beginning from the seed with size 0, the algorithm repeats to find one-size larger seeds until the size reaches $d_{\mathcal{C}} + 1$. Note that it is not needed to give $d_{\mathcal{C}}$ to the algorithm because the algorithm detects $d_{\mathcal{C}} + 1$ automatically. Using this algorithm, concepts with small characteristic dimension are guaranteed to be found in early stages. The algorithm runs in $O((d_{\mathcal{C}} + 1)|\mathcal{C}|n^{d_{\mathcal{C}}+1})$ time, where n is the number of elements in a domain.

In EnumSpernerFamily, a subconcept query is asked for each seed candidate, and a seed candidate is a subset which satisfies the condition that it is a transversal of compliments of the concepts found so far. Thus, EnumSpernerFamily can be seen as an instance of algorithm ALL_MSS developed by Gunopulos et al. [3], an instance that uses a new algorithm called EnumMinTrans for calculating the transversal hypergraph of $\overline{\mathcal{C}}$, where $\mathcal{C}$ is a target Sperner concept class. The computational bottle neck of algorithm ALL_MSS is the calculation of a transversal hypergraph, and incremental algorithms [5, 3, 6] are considered to be efficient in practice. The most sophisticated instance of ALL_MSS using such an incremental algorithm is *Irredundant Border Enumerator* (IBE) developed by Uno and Satoh [6]. We show that IBE is also guaranteed to be efficient when both $|\mathcal{C}|$ and $d_{\mathcal{C}}$ are small by proving its computational time bound $O((|\mathcal{C}|n^{d_{\mathcal{C}}+2})$, which is larger than that of EnumSpernerFamily by factor n.

2 Preliminary

A subset of a set X is called a *concept* on X, and a family of concepts on X is called a *concept class* on X. A concept class $\mathcal{C}$ is a *Sperner family* if no element is contained in another. Formally,

$$C_1, C_2 \in \mathcal{C} \text{ and } C_1 \neq C_2 \Rightarrow C_1 \not\subset C_2 \text{ and } C_1 \not\supset C_2.$$

Note that X and $\mathcal{C}$ correspond to sets of vertices and hyperedges, respectively, in the context of hypergraph. A hypergraph $(X, \mathcal{C})$ with Sperner family $\mathcal{C}$ is said to be *simple*. A hypergraph $(X, \mathcal{C})$ is also denoted by its hyperedge set $\mathcal{C}$ only throughout this paper. We assume that any concept class considered in this paper is non-empty.

We assume existence of a teacher (or oracle) who can answer *subconcept queries* which ask if a subset T of X is contained in some concept C belonging to a target concept class $\mathcal{C}$. The answers of the teacher is 'yes' or 'no'. In this paper, we consider the following problem using this query studied by Gunopulos et al. [3]. (The problem is called Problem MaxTh in [3].)

Problem 1 *Enumerate concepts in an unknown Sperner family concept class $\mathcal{C}$ on X using subconcept queries for $\mathcal{C}$.*

In the followings, we describe notions and a proposition necessary in this paper. A concept class $\mathcal{C}$ is *intersection closed* if $\bigcap_{C \in \mathcal{U}} C \in \mathcal{C}$ holds for every non-empty subfamily $\mathcal{U} \subseteq \mathcal{C}$. Let $\mathcal{C}$ be a concept class on X. For an arbitrary subset S of X, define a concept class $\Pi_{\mathcal{C}}(S)$ on S as follows:

$$\Pi_{\mathcal{C}}(S) = \{S \cap C : C \in \mathcal{C}\}.$$

Set S is said to be *shattered by* $\mathcal{C}$ if $\Pi_{\mathcal{C}}(S) = 2^S$ holds. The VC dimension [1] of $\mathcal{C}$ is the size of a maximum-sized set shattered by $\mathcal{C}$, where the size of a set is the number of elements in a set.

The next proposition holds for intersection closed concepts.

Proposition 2 *Let $\mathcal{C}$ be an intersection closed concept class. Then, a subset S of X is shattered by $\mathcal{C}$ if and only if the following conditions (1) and (2) hold:*

(1) $S \subseteq C$ for some $C \in \mathcal{C}$ and,

(2) for all $x \in S$, there exists $C \in \mathcal{C}$ such that $S - \{x\} \subseteq C$ and $x \notin C$.

Proof: It is trivial that conditions (1) and (2) hold if S is shattered by $\mathcal{C}$. We prove the opposite direction. Assume that conditions (1) and (2) hold. For each $x \in S$, let C_x denote the concept C that satisfies condition (2). Then, for an arbitrary proper subset S' of S,

$$S \cap \bigcap_{x \in S - S'} C_x = \bigcap_{x \in S - S'} (S - \{x\}) = S'$$

holds. Note that $\bigcap_{x\in S-S'} C_x \in \mathcal{C}$ holds because $\mathcal{C}$ is intersection closed. This, together with condition (1), proves that S is shattered by $\mathcal{C}$. ■

3 Complexity of Sperner Family Concept Classes

3.1 VC Dimension and Characteristic Dimension

In this subsection, we characterize complexity of Sperner family concept classes by two measures, the VC dimension and the *characteristic dimension*.

First, let us consider characterization by the VC dimension. We do NOT directly characterize a Sperner family concept class $\mathcal{C}$ by its VC dimension but characterize it by the VC dimension of its intersection closure $\mathrm{cl}(\mathcal{C}) = \{\bigcap_{C\in\mathcal{U}} C : \emptyset \neq \mathcal{U} \subseteq \mathcal{C}\}$.

Since $\mathrm{cl}(\mathcal{C})$ is intersection closed, the following corollary of Proposition 2 holds.

Corollary 3 *Let $\mathcal{C}$ be a Sperner family concept class on X. Then, a subset T of X is shattered by $cl(\mathcal{C})$ if and only if the following two conditions (1) and (2) hold for $\mathcal{C}$:*

(1) $T \subseteq C$ for some $C \in \mathcal{C}$, and

(2) for each $x \in T$, there exists $C \in \mathcal{C}$ such that $T - \{x\} \subseteq C$ and $x \notin C$.

Next, we give the definition of *characteristic dimension*. Let $\mathcal{C}$ be a Sperner family concept class on X. For each $C \in \mathcal{C}$, a subset T of X satisfying the following two properties is called a *characteristic set* of C.

Property 1 C is an unique member of $\mathcal{C}$ that contains T.

Property 2 for each $x \in T$, there exists $C_x \in \mathcal{C}$ such that $T - \{x\} \subseteq C_x$ and $C_x \neq C$.

Note that $\emptyset$ is a characteristic set of $\mathcal{C}$ that is composed of one concept only. For each concept $C \in \mathcal{C}$, there exists a characteristic set T of C because of the following reason. First, let $T = C$. Note that T satisfies Property 1 trivially. Remove as many elements as possible from T while Property 1 is satisfied. Then, the contracted T must satisfy Property 2. There may exist more than one characteristic sets for one concept and the sizes of those may be different.

Example 4 *Let X be $\mathbb{R}^2$, and consider a set of 10 positive elements $\{1,2,3,4,5,6,7,8,9,10\}$ and 4 negative elements shown in Figure 1. Let $\mathcal{C}$ be the family of maximal positive-element sets that are contained in some axis-parallel rectangles which contains no negative instances. Trivially, $\mathcal{C}$ is a Sperner family composed of 5 concepts $\{1,3,4,5,7\}, \{2,3,4,5,6\}, \{3,4,5,6,7,8\}, \{5,6,7,8,9\}$ and $\{4,6,7,8,10\}$. The characteristic sets of a concept $\{3,4,5,6,7,8\}$ are $\{3,8\}, \{3,6,7\}, \{4,5,6,7\}$ and $\{4,5,8\}$.*

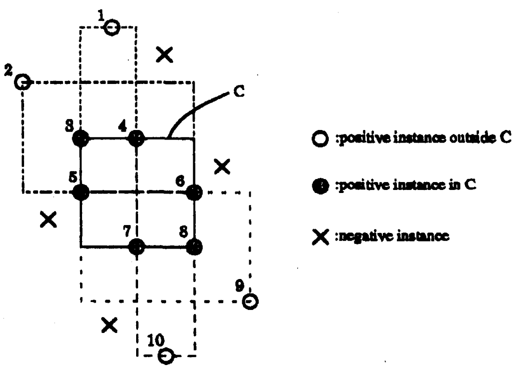

Figure 1: Example of characteristic sets

The *characteristic dimension* of a concept C is the number of elements in the minimum characteristic sets of C. The maximum characteristic dimension of the concepts in a Sperner family concept class $\mathcal{C}$ is the *characteristic dimension* of $\mathcal{C}$.

Proposition 5 *For an arbitrary Sperner family concept class $\mathcal{C}$, the characteristic dimension of $\mathcal{C}$ is at most the VC dimension of $cl(\mathcal{C})$.*

Proof: Let T be a minimum characteristic set of C in $\mathcal{C}$. Then, Property 1 and 2 hold for T, so Corollary 3 (1) and (2) hold for this T. Therefore, T is shattered by cl($\mathcal{C}$). This means that the characteristic dimension of $\mathcal{C}$ is at most the VC dimension of cl($\mathcal{C}$). ■

Remark 6 *The characteristic dimension of a Sperner family concept class $\mathcal{C}$ can be smaller than the VC dimension of cl($\mathcal{C}$). Actually, the characteristic dimension of $\mathcal{C}$ in the previous example (Figure 1) is 2 while the VC dimension of cl($\mathcal{C}$) is 4.*

3.2 Relation to the Rank of the Minimal Transversal Hypergraph

Let $\mathcal{C}$ be a Sperner family concept class in X. A set $T \subseteq X$ is a *transversal* of hypergraph $\mathcal{C}$ if $T \cap C \neq \emptyset$ for each $C \in \mathcal{C}$. The *transversal hypergraph* Tr($\mathcal{C}$) of hypergraph $\mathcal{C}$ is a hypergraph whose edges are the minimal transversals of $\mathcal{C}$. The *rank* $r(\mathcal{C})$ of hypergraph $\mathcal{C}$ is defined as the maximum set size in $\mathcal{C}$, namely, $r(\mathcal{C}) = \max_{C \in \mathcal{C}} |C|$, where $|C|$ is the number of elements in C. The compliment set of C is denoted by $\overline{C}$, and $\overline{\mathcal{C}}$ denotes the set of the compliment sets of Cs in $\mathcal{C}$.

Proposition 7 *Let $\mathcal{C}$ be a Sperner family concept class on X. Then, if T is a minimal transversal of $\overline{\mathcal{C}}$, $T - \{x_0\}$ is shattered by cl($\mathcal{C}$) for $x_0 \in T$.*

Proof: Let T be a minimal transversal of $\overline{\mathcal{C}}$ and let x_0 be an element of T. From the minimality of T, for each $x \in T$, there is a set $\overline{C} \in \overline{\mathcal{C}}$ with $\overline{C} \cap T = \{x\}$. This means that there is a set $C \in \mathcal{C}$ with $T - \{x\} \subseteq C$ and $x \notin C$. Then, conditions (1) and (2) of Corollary 3 hold for $T - \{x_0\}$. ■

Corollary 8 *Let $d_{\mathcal{C}}$ be the VC dimension of cl($\mathcal{C}$). Then, $r(Tr(\overline{\mathcal{C}})) \leq d_{\mathcal{C}} + 1$.*

Remark 9 *From Corollary 8, we know the VC dimension of cl($\overline{\mathcal{H}}$) plus 1 is an upper bound of the rank of the transversal hypergraph of $\mathcal{H}$. Is this upper bound always tight? Unfortunately, the answer is no. Consider hypergraph $\mathcal{H} = \{S_0, S_1, ..., S_k\}$, where $S_0 = \{1, 2, ..., k\}$ and $S_i = S_0 \cup \{k+i\} - \{i\}$. In this case, the VC dimension of cl($\overline{\mathcal{H}}$) is k while $r(Tr(\mathcal{H}))$ is only 2. Note that characteristic dimension of $\overline{\mathcal{H}}$ is also k. Let $\mathcal{H}' = \mathcal{H} - \{S_0\}$. Then, the VC dimension of cl($\overline{\mathcal{H}'}$) is $k-1$, and $r(Tr(\mathcal{H}'))$ is k, so the bound is tight. In this case, the characteristic dimension of $\overline{\mathcal{H}'}$ is only 1. This phenomenon occurs because the VC dimension and characteristic dimension always increases when new concept (hyperedge) is added but this is not true as for the rank of the transversal hypergraph. Furthermore, addition of one new concept changes the VC dimension by at most one, but it may drastically change the characteristic dimension and the rank of the transversal hypergraph.*

4 Enumeration Algorithm

4.1 Algorithm EnumSpernerFamily

In this subsection, we describe a new algorithm of enumerating concepts in a Sperner family concept class using subconcept queries, and prove its correctness and an upper bound of its time complexity.

EnumSpernerFamily shown in Figure 2 is an algorithm of repeatedly executing the procedure that finds a seed (subset) of a concept which has not been found yet and obtains one concept by making the seed grow up using subroutine FindOneConcept[1]. The first seed is the empty set and one concept C_0 in the target Sperner family concept class $\mathcal{C}^*$ can be found by the assumption that $\mathcal{C}^* \neq \emptyset$. In the algorithm, i is the seed size and increased from 0 until no seed with size i satisfies the condition (1) and (2) in Corollary 3. Note that SubCon(T) in the algorithm denotes a subconcept query which asks if T is contained in some concept C belonging to $\mathcal{C}^*$, and the value of SubCon(T) is the answer of the question, namely, 'true' if contained or 'false' otherwise.

[1]FindOneConcept is the same algorithm as algorithm AMSS used by Gunopulos et al. [3].

```
EnumSpernerFamily
output: $\mathcal{C}$ : Sperner family concept class on $X$
        equal to an unknown target $\mathcal{C}^*$.
1: $\mathcal{C} \leftarrow \emptyset$
2: $C_0 \leftarrow$ FindOneConcept($\emptyset$)
3: $\mathcal{C} \leftarrow \mathcal{C} \cup \{C_0\}$
4: $i \leftarrow 0$
5: while there exists an $i$-element subset $T$ of
   $X$ that satisfies the conditions (1) and (2) in
   Corollary 3 do
6:   $i \leftarrow i + 1$
7:   for all $T$: $i$-element subsets of $X$ that satisfy
     the condition (2) in Corollary 3 but do NOT
     satisfy the condition (1) in the corollary do
8:     if SubCon($T$) then
9:       $C \leftarrow$ FindOneConcept($T$)
10:      $\mathcal{C} \leftarrow \mathcal{C} \cup \{C\}$
11:    end if
12:  end for
13: end while
14: output $\mathcal{C}$
```

```
FindOneConcept($T$)
input: $T$: subset of some concept in $\mathcal{C}^*$
output: $C$: one concept in $\mathcal{C}^*$.
1: $C \leftarrow T$
2: for each $x \in X$ do
3:   if SubCon($C \cup \{x\}$) then
4:     $C \leftarrow C \cup \{x\}$
5:   end if
6: end for
7: output $C$
```

Figure 2: Algorithm EnumSpernerFamily

Lemma 10 *For any $T \subseteq X$, at least one of the two conditions (1) and (2) in Corollary 3 become satisfied before the for-loop with $i = |T|$ in EnumSpernerFamily if there exists a concept $C \in \mathcal{C}^*$ that contains T.*

Proof: Assume that both the conditions are NOT satisfied at right before the for-loop with $i = |T|$ in the algorithm though there exists a concept in $\mathcal{C}^*$ contains T. Consider $\mathcal{C}$ at right before the for-loop[2] with $i = |T|$. Since the condition (1) in Corollary 3 is not satisfied, any concept $C \in \mathcal{C}^*$ including T is not contained in $\mathcal{C}$. There exists an $x \in T$ such that no concept $C_x \in \mathcal{C}$ excluding x contains $T - \{x\}$ by the assumption that the condition (2) in Corollary 3 is not satisfied. These two facts imply that no concept including $T - \{x\}$ is not contained in $\mathcal{C}$.

Now, redefine T as $T - \{x\}$, a set of one element smaller. Then, even for this redefined T, both the conditions are NOT satisfied at right before the for-loop with $i = |T|$ by the following reason. The condition (1) in Corollary 3 is not satisfied trivially. Let us assume that the condition (2) in Corollary 3 is satisfied. Then, when the for-loop with $i = |T|$ is executed, $\mathcal{C}$ must be updated so as to contain some concept C containing this redefined T at the for-loop with $i = |T|$, but this contradicts the fact. On the other hand, when the algorithm stops without executing the for-loop with $i = |T|$, consider $\mathcal{C}$ at the end of the algorithm. For this $\mathcal{C}$, the condition (2) cannot not be satisfied because the assumption of satisfying the condition (2) implies satisfaction of the while-loop condition for all $i \leq |T|$, which contradicts the fact the algorithm has already stopped.

By repeating this argument, there exists a further redefined $T = \{x\}$ such that both the conditions are not satisfied right before the for-loop with $i = 1$. Then, $\mathcal{C}$ must be $\emptyset$, which contradicts the assumption that $\mathcal{C}^*$ is non-empty.

[2]When EnumSpernerFamily stops without executing the for-loop with $i = |T|$, consider $\mathcal{C}$ at the end of the algorithm.

Therefore, at least one of the two conditions (1) and (2) in Corollary 3 become satisfied before the for-loop with $i = |T|$ for the original T. ■

Theorem 11 *Let C be a concept of a Sperner family concept class $\mathcal{C}^*$. Then, C is added to $\mathcal{C}$ in algorithm EnumSpernerFamily until the end of the for-loop with $i = d_C$, where d_C be the characteristic dimension of C.*

Proof: Let T be a characteristic set of C that is composed of just d_C elements. Note that C is the unique concept that contains T. Then, by Lemma 10, at least one of two conditions (1) and (2) become satisfied before the for-loop with $i = d_C$. In the case that the condition (1) is satisfied, C has been already included in $\mathcal{C}$. In the case that the condition (1) is not but the condition (2) is satisfied, C is added to $\mathcal{C}$ during the execution of the for-loop with $i = d_C$. Thus, C is added to $\mathcal{C}$ until the end of the for-loop with $i = d_C$. ■

Remark 12 *The algorithm cannot be stopped even if no concept is found in the for-loop. For example, consider a set of 8 positive points $\{1,2,3,4,5,6,7,8\}$ and 4 negative points on $\mathbb{R}^2$ shown in Figure 3. Let $\mathcal{C}^*$ be the family of maximal positive-point sets that are contained in some axis-parallel rectangles which contains no negative instances. $\mathcal{C}^*$ is composed of 5 concepts $\{1,2,4,7\}, \{2,3,4,5\}, \{2,4,5,7\}, \{2,5,7,8\}$ and $\{4,5,6,7\}$. Four concepts $\{1,2,4,7\}, \{2,3,4,5\}, \{2,5,7,8\}$ and $\{4,5,6,7\}$ having characteristic dimension 1 are found by the end of the for-loop with $i = 1$, but the concept $\{2,4,5,7\}$ with characteristic dimension 4 might not be found until the seed size i becomes 4.*

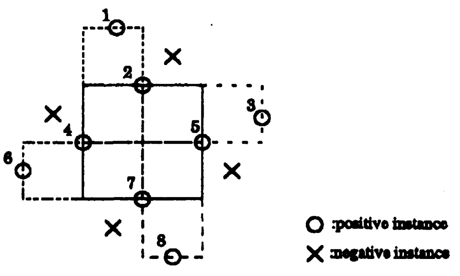

Figure 3: Example of the case that a concept is possibly found after the for-loop in which no concept is found.

Corollary 13 *EnumSpernerFamily outputs a concept class $\mathcal{C}$ equal to $\mathcal{C}^*$. Besides, it stops just after executing the for-loop with $i = d_{\mathcal{C}^*} + 1$, where $d_{\mathcal{C}^*}$ is the VC dimension of $cl(\mathcal{C}^*)$.*

Proof: By Theorem 11, all the concepts in $\mathcal{C}^*$ are added to $\mathcal{C}$ in the algorithm. Thus, the outputted $\mathcal{C}$ is trivially equal to $\mathcal{C}^*$. The procedure inside the while-loop is executed for $i+1$ while i-element set is shattered by $\mathcal{C}$. Therefore, the algorithm stops just after executing the for-loop with $i = d_{\mathcal{C}^*} + 1$. ■

Theorem 14 *EnumSpernerFamily runs in time $O(d_{\mathcal{C}^*} + 1)|\mathcal{C}^*|n^{d_{\mathcal{C}^*}+1})$, where $d_{\mathcal{C}^*}$ is the VC dimension of $cl(\mathcal{C}^*)$, and $n = |X|$.*

Proof: By Corollary 13, the number of subset T of X checked in the for-loop whether the conditions in Corollary 3 are satisfied or not is at most $n^{d_{\mathcal{C}^*}+1}$ in total. This check can be done in time $O(|T||\mathcal{C}|)$ for each T and $\mathcal{C}$ at that point by keeping a sorted id list of found concepts containing an element x for each x in X. Thus, this is upper bounded by $O((d_{\mathcal{C}^*}+1)|\mathcal{C}^*|)$. Total time needed by execution of the procedure FindOneConcept is $O(n|\mathcal{C}^*|)$ trivially. By putting all together, we obtain the bound $O(d_{\mathcal{C}^*} + 1)|\mathcal{C}^*|n^{d_{\mathcal{C}^*}+1})$. ■

4.2 Relation to Algorithm ALL_MSS

Proposition 15 *For a subset T of X and a hypergraph $\mathcal{C}$ with vertex set X, condition (1) of Corollary 3 does NOT hold but its condition (2) does hold if and only if T is a minimal transversal of $\overline{\mathcal{C}}$.*

Proof: Assume that condition (1) of Corollary 3 does NOT hold but its condition (2) does hold. Then, T is a transversal of $\overline{\mathcal{C}}$ because condition (1) does not hold. T is also minimal because from condition (2), for each $x \in T$, there exists $C \in \mathcal{C}$ such that $(T - \{x\}) \cap \overline{C} = \emptyset$. Conversely, assume that T is a minimal transversal of $\overline{\mathcal{C}}$. T is a traversal so condition (1) does not hold. Condition (2) holds because there is a $\overline{C} \in \overline{\mathcal{C}}$ such that $T \cap \overline{C} = \{x\}$ for each $x \in T$ from the minimality of T. ■

By the above proposition, T that satisfies the condition of line 7 in algorithm EnumSpernerFamily is a minimal transversal of $\overline{\mathcal{C}}$. This fact means that EnumSpernerFamily can be seen as a kind of algorithm ALL_MSS developed by Gunopulos et al. [3]. Their algorithm uses the algorithm for enumerating the minimal transversals of $\overline{\mathcal{C}}$ as a subroutine, and an algorithm similar to ours is derived from their algorithm using algorithm EnumMinTrans shown in Figure 4 as the subroutine.

```
EnumMinTrans($X, \mathcal{H}$)
input: $(X, \mathcal{H})$ : hypergraph
output: $\mathcal{G}$ : transversal hypergraph of $\mathcal{H}$
1: $\mathcal{G} \leftarrow \emptyset$
2: $i \leftarrow 0$
3: while there exist an $i$-element subset $T$ of $X$ that satisfies
   the conditions (1) and (2) in Corollary 3 for $\mathcal{C} = \overline{\mathcal{H}}$ do
4:   $i \leftarrow i + 1$
5:   for all $T$: $i$-element subsets of $X$ that satisfy the condition (2) in Corollary 3 but do NOT satisfy the condition
     (1) in the corollary for $\mathcal{C} = \overline{\mathcal{H}}$ do
6:     $\mathcal{G} \leftarrow \mathcal{G} \cup \{T\}$
7:   end for
8: end while
9: output $\mathcal{G}$
```

Figure 4: Algorithm EnumMinTrans

All the minimal transversals of $\mathcal{H}$ are enumerated by algorithm EnumMinTrans because, at the exit of the while-loop, i is larger than VC dimension of $\mathrm{cl}(\overline{\mathcal{H}})$, and this fact induces that $r(Tr(\mathcal{H})) \leq i$ using Corollary 8 while all minimal transversals with size at most i are already enumerated by that time. EnumMinTrans is practically useful when the VC dimension of $\mathrm{cl}(\overline{\mathcal{H}})$ is small. The computational time of EnumMinTrans is $O((d_{\overline{\mathcal{H}}} + 1)|\mathcal{H}|n^{d_{\overline{\mathcal{H}}}+1})$ from the analysis in the proof of Theorem 14, where $d_{\overline{\mathcal{H}}}$ is the VC dimension of $\mathrm{cl}(\overline{\mathcal{H}})$, and $n = |X|$.

One more merit of EnumMinTrans is independency from the already found transversals. So the algorithm can forget the previously found transversals if they are not needed finally. EnumMinTrans is not directly used by EnumSpernerFamily but embedded into it in a way that EnumMinTrans is executed only once[3] and as a result no set T is reproduced.

Remark 16 *Note that not all transversals of $\overline{\mathcal{C}}$ are seeds of concepts $C \notin \mathcal{C}$. Indeed, Theorem 11 means that transversals with the size at most the characteristic dimension of $\mathcal{C}^*$ are enough for the seeds. As mentioned in Remark 9, the characteristic dimension of $\mathcal{C}^*$ is sometimes smaller than $r(Tr(\overline{\mathcal{C}^*}))$, so in such a case EnumSpernerFamily might become more efficient if it could exit its while-loop at the end of the for-loop with $i = d$, where d is the characteristic dimension of $\mathcal{C}^*$.*

[3]Precisely speaking, this is not true because $\mathcal{C}$ is changing in EnumSpernerFamily while $\mathcal{C} = \overline{\mathcal{C}}$ is fixed in EnumMinTrans. However, the computational cost for such changing $\mathcal{C}$ is trivially smaller than that for fixed $\mathcal{C}^*$.

4.3 Analysis of Algorithm IBE

We showed that Algorithm EnumSpernerFamily is one implementation of algorithm ALL_MSS with some merits. There already exists such an algorithm that has the same merits, Irredundant Border Enumerator (IBE) developed by Uno and Satoh [6]. In this subsection, we analyze the algorithm and induce an similar upper bound on its computational time in terms of the VC dimension $d_{\mathcal{C}}$ of intersection closure of a Sperner family concept class $\mathcal{C}$.

As mentioned above, ALL_MSS has a freedom for the choice of an algorithm for calculating the traversal hypergraph $Tr(\overline{\mathcal{C}})$. The fastest algorithm for the task theoretically is the one developed by Fredman and Khachiyan [2], and its time complexity is $(|\mathcal{C}| + |Tr(\overline{\mathcal{C}})|)^{o(\log(|\mathcal{C}|+|Tr(\overline{\mathcal{C}})|))}$. But this algorithm is not considered to be practical because of its heavy memory use and difficulty of implementation. The most appropriate conventional algorithms for calculating $Tr(\overline{\mathcal{C}})$ used in ALL_MSS are incremental ones [5, 3, 6]. Here, an incremental algorithm for calculating $Tr(\overline{\mathcal{C}})$ means that the algorithm first calculates $Tr(\overline{\mathcal{C}}_1)$, next $Tr(\overline{\mathcal{C}}_2)$,..., and finally $Tr(\overline{\mathcal{C}}_{|\mathcal{C}|})$, where $\overline{\mathcal{C}}_i = \{\overline{C}_1, \overline{C}_2, ..., \overline{C}_i\}$ and $\overline{\mathcal{C}} = \overline{\mathcal{C}}_{|\mathcal{C}|}$. Such incremental algorithms are easy to implement, and they are suited for ALL_MSS because concepts in $\mathcal{C}$ are found one by one in ALL_MSS. In the worst case, those algorithms are known to possibly spend time exponential to $|\mathcal{C}| + |Tr(\overline{\mathcal{C}})|$ because $|Tr(\overline{\mathcal{C}}_i)|$ for some $i < |\mathcal{C}|$ may become exponential. However, by Corollary 8, $r(Tr(\overline{\mathcal{C}}_i)) \leq d_{\mathcal{C}_i} + 1 \leq d_{\mathcal{C}} + 1$ holds, namely, $Tr(\overline{\mathcal{C}}_i) \subseteq X^{d_{\mathcal{C}}+1}$ for all $i < |\mathcal{C}|$, where $d_{\mathcal{C}_i}$ and $d_{\mathcal{C}}$ are the VC dimensions.of $cl(\{C_1, C_2, ..., C_i\})$ and $cl(\mathcal{C})$, respectively. This means that incremental algorithms are efficient when VC dimension of $cl(\mathcal{C})$ is small. In IBE, a kind of algorithm ALL_MSS using the most efficient incremental algorithm for calculating $Tr(\overline{\mathcal{C}})$, enumeration of hyperedges in $\bigcup_{i \leq |\mathcal{C}|} Tr(\overline{\mathcal{C}}_i)$ is done by the depth first transversal of a tree whose nodes correspond to hyperedges in $\bigcup_{i \leq |\mathcal{C}|} Tr(\overline{\mathcal{C}}_i)$, and computational time needed at each node is $O(n|\mathcal{C}|)$, where $n = |X|$. Since this enumeration is the bottle neck of IBE, its total running time is $O(|\mathcal{C}|n^{d_{\mathcal{C}}+2})$. Regarding $d_{\mathcal{C}}$ as a constant, this bound is worse than that of EnumSpernerFamily by factor n, but IBE seems more efficient practically because $|\bigcup_{i \leq |\mathcal{C}|} Tr(\overline{\mathcal{C}}_i)|$ is significantly smaller than $n^{d_{\mathcal{C}}+1}$ in most cases.

Acknowledgements

We would like to thank Prof. Takeaki Uno of National Institute of Informatics for his helpful comments.

References

[1] A. Blumer, A. Ehrenfeucht, D. Haussler and M. Warmuth, Learnability and the Vapnik-Chervonenkis Dimension, *Journal of the ACM* 36(4), pp.929-965, 1989.

[2] M. Fredman and L. Khachiyan, On the Complexity of Dualization of Monotone Disjunctive Normal Forms, *Journal of Algorithms*, 21, pp.618-628, 1996.

[3] D. Gunopulos, R. Khardon, H. Mannila, S. Saluja, H. Toivonen and R. Sharma, Discovering All Most Specific Sentences, *ACM Transactions on Database Systems*, 28(2), PP.140-174, 2003.

[4] K. Gouda and M. Zaki, GenMax: An Efficient Algorithm for Mining Maximal Frequent Itemsets, *Data Mining and Knowledge Discovery* 11, pp.1-20, 2005.

[5] D. Kavvadias and E. Stavropoulos, Evaluation of an Algorithm for the Transversal Hypergraph Problem, *Third International Workshop on Algorithm Engineering*, pp.72-84, 1999.

[6] T. Uno and K.Satoh, Detailed Description of an Algorithm for Enumeration of Maximal Frequent Sets with Irredundant Dualization, FIMI03.