

Concise Descriptions of Subsets of Structured Sets

Alberto O. Mendelzon
University of Toronto
mendel@cs.toronto.edu

Ken Q. Pu
University of Toronto
kenpu@cs.toronto.edu

ABSTRACT

We study the problem of economical representation of subsets of structured sets, that is, sets equipped with a set cover. Given a structured set U , and a language \mathcal{L} whose expressions define subsets of U , the problem of Minimum Description Length in \mathcal{L} (\mathcal{L} -MDL) is: “given a subset V of U , find a shortest string in \mathcal{L} that defines V ”.

We show that the simple set cover is enough to model a number of realistic database structures. We focus on two important families: hierarchical and multidimensional organizations. The former is found in the context of semistructured data such as XML, the latter in the context of statistical and OLAP databases. In the case of general OLAP databases, data organization is a mixture of multidimensionality and hierarchy, which can also be viewed naturally as a structured set. We study the complexity of the \mathcal{L} -MDL problem in several settings, and provide an efficient algorithm for the hierarchical case.

Finally, we illustrate the application of the theory to summarization of large result sets, (multi) query optimization for ROLAP queries, and XML queries.

1. INTRODUCTION

Consider an OLAP or multidimensional database setting [5], where a user has requested to view a certain set of cells of the datacube, say in the form of a 100 x 20 matrix. Typically, the user interacts with a front-end query tool that ships SQL queries to a back end dbms. After perusing the output, the user clicks on some of the rows of the matrix, say 20 of them, and requests further details on these rows. Suppose each row represents data on a certain city. A typical query tool will translate the user request to a long SQL query with a WHERE clause of the form `city = city1 OR city = city2 ... OR city = city20`. However, if the set of cities happens to include every city in Ontario except Toronto, an equivalent but much shorter formulation would be `province = 'Ontario' AND city <> 'Toronto'`. Minimizing the length of the query that goes to the back end

is advantageous for two reasons. First, many systems have difficulty dealing with long queries, or even hard limits on query length. Second, the shorter query can often be processed much faster than the longer one (even though an extra join may be required, e.g. if there is no `Province` attribute stored in the cube.)

With this problem as motivation, we study the concise representations of subsets of a structured set. By “structured” we simply mean that we are given a (finite) set, called the universe, and a (finite) set of symbols, called the alphabet, each of which represents some subset of the universe. We are also given a language \mathcal{L} of expressions on the alphabet, and a semantics that maps expressions to subsets of the universe. Given a subset V of the universe, we want to find a shortest expression in the given language that describes V . We call this the \mathcal{L} -MDL (*Minimum Description Length*) problem. In the example above, the universe is the set of city names, the alphabet includes at least the city name `Toronto` plus a set of province names, and the semantics provides a mapping from province names to sets of cities. This is the simplest case, where the symbols in the alphabet induce a partition of the universe.

The most general language we consider, called \mathcal{L} , is the language of arbitrary Boolean set expressions on symbols from the alphabet. In Section 3 we show that the \mathcal{L} -MDL problem is solvable in polynomial time when the alphabet forms a partition of the universe. In particular, when the partition is *granular*, that is, every element of the universe is represented as one of the symbols in the alphabet, we obtain a normal form for minimum-length expressions, leading to a polynomial time algorithm.

Of course, in addition to cities grouped into provinces, we could have provinces grouped into regions, regions into countries, etc. That is, the subsets of the universe may form a *hierarchy*. We consider this case in Section 4 and show that the normal forms of the previous section can be generalized, leading again to a polynomial time \mathcal{L} -MDL problem.

In the full OLAP context, elements of the universe can be grouped according to multiple independent criteria. If we think of a row in our initial example as a tuple `<city,product,date,sales>`, and the universe is the set of such tuples, then these tuples can be grouped by city into provinces, or by product into brands, or by date into years, etc. In Section 5 we consider the multidimensional case. In particular, we focus on the common situation in which each of the groupings is a hierarchy. We consider three increasingly powerful sublanguages of \mathcal{L} , including \mathcal{L} itself, and show that the MDL problem is NP-complete for

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

PODS 2003, June 9-12, 2003, San Diego, CA.

Copyright 2003 ACM 1-58113-670-6/03/06 ...\$5.00.

each of them.

In Section 6 we discuss in more detail the application of our formal results to the OLAP setting, both when optimizing a single query and a set of related queries. We also present another application, to the problem of minimizing query length for queries on structured documents such as those written in XML.

2. STRUCTURED SETS, LANGUAGES AND THE MDL PROBLEM

In this section we introduce our model of structured sets and descriptive languages for subsets of them, and state the Minimum Description Length problem.

DEFINITION 1 (STRUCTURED SET). *A structured set is a pair of finite sets (U, Σ) together with an interpretation function $[\cdot] : \Sigma \rightarrow \mathbf{Pwr}(U) : u \mapsto [u]$ which is injective. The set U is referred as the universe, and Σ the alphabet.*

Intuitively the structure of the set U is modeled by the grouping of its elements; each group is labeled by a symbol in the alphabet Σ . The interpretation of a symbol σ is the elements in U belonging to the group labeled by σ .

Elements of the alphabet can be combined in expressions that describe other subsets of the universe. The most general language we will consider for these expressions is the *propositional language*, that consists of all expressions composed of symbols from the alphabet and operators that stand for the usual set operations of union, intersection and difference.

DEFINITION 2 (PROPOSITIONAL LANGUAGE). *Given a structured set (U, Σ) , its propositional language $\mathcal{L}(U, \Sigma)$ is defined as: $\epsilon \in \mathcal{L}(U, \Sigma)$, $\sigma \in \mathcal{L}(U, \Sigma)$ for all $\sigma \in \Sigma$, and if $\alpha, \beta \in \mathcal{L}(U, \Sigma)$, then $(\alpha + \beta)$, $(\alpha - \beta)$ and $(\alpha \cdot \beta)$ are all in $\mathcal{L}(U, \Sigma)$.*

DEFINITION 3 (SEMANTICS AND LENGTH). *The evaluation of $\mathcal{L}(U, \Sigma)$ is a function $[\cdot]^* : \mathcal{L}(U, \Sigma) \rightarrow \mathbf{Pwr}(U)$, defined as: $[\epsilon]^* = \emptyset$, $[\sigma]^* = [\sigma]$ for any $\sigma \in \Sigma$, and $[\alpha + \beta]^* = [\alpha]^* \cup [\beta]^*$, $[\alpha - \beta]^* = [\alpha]^* - [\beta]^*$, $[\alpha \cdot \beta]^* = [\alpha]^* \cap [\beta]^*$.*

The string length of $\mathcal{L}(U, \Sigma)$ is a function $\|\cdot\| : \mathcal{L}(U, \Sigma) \rightarrow \mathbb{N}$, given by: $\|\epsilon\| = 0$, $\|\sigma\| = 1$ for any $\sigma \in \Sigma$, and $\|\alpha + \beta\| = \|\alpha - \beta\| = \|\alpha \cdot \beta\| = \|\alpha\| + \|\beta\|$.

Remark: We abuse the definitions in a number of harmless ways. For instance, we may refer to U as a structured set, implying that it is equipped with an alphabet Σ and an interpretation function $[\cdot]$. The language $\mathcal{L}(U, \Sigma)$ is sometimes written simply as \mathcal{L} when the structured set (U, Σ) is understood from the context. The evaluation function $[\cdot]^*$ supersedes the single-symbol interpretation function $[\cdot]$, so the latter is omitted from discussions and the simpler form $[\cdot]$ is used in place of $[\cdot]^*$.

Two expressions s and t in \mathcal{L} are *equivalent* if they evaluate to the same set: i.e. $[s] = [t]$. (Note that this means equivalence with respect to a particular structured set (U, Σ) and thus does not coincide with propositional equivalence). In case they are equivalent, we say that s is *reducible* to t if $\|s\| \geq \|t\|$. The expression s is *strictly reducible* to t if they are equivalent and $\|s\| > \|t\|$. An expression is *compact* if it is not strictly reducible to any other expression in the language.

Given a sub-language $\mathcal{K} \subseteq \mathcal{L}$, an expression is *\mathcal{K} -compact* if it is not strictly reducible to any other expression in \mathcal{K} .

A language $\mathcal{K} \subseteq \mathcal{L}(U, \Sigma)$ is *granular* if it can express every subset, or equivalently, every singleton, i.e.

$$\forall a \in U. \exists s \in \mathcal{K}. [s] = \{a\}.$$

We say that a structure Σ is *granular* if the propositional language $\mathcal{L}(U, \Sigma)$ is granular.

If $\mathcal{L}(U, \Sigma)$ is not granular, then certain subsets (specifically singletons) of U cannot be expressed by any expression. The solution is then to augment the alphabet Σ to include sufficiently more symbols until it becomes granular.

DEFINITION 4 (\mathcal{K} -DESCRIPTIVE LENGTH). *Given a structured set (U, Σ) , consider a sub-language $\mathcal{K} \subseteq \mathcal{L}(U, \Sigma)$. The \mathcal{K} -descriptive length of a subset $V \subseteq U$, written $\|V\|_{\mathcal{K}}$ is defined as,*

$$\|V\|_{\mathcal{K}} = \begin{cases} \min\{\|\alpha\| : \alpha \in \mathcal{K}(V)\} & \text{if } \mathcal{K}(V) \neq \emptyset, \text{ and} \\ \infty & \text{otherwise.} \end{cases}$$

In case $\mathcal{K} = \mathcal{L}(U, \Sigma)$, we write $\|V\|_{\mathcal{K}}$ simply as $\|V\|$.

The \mathcal{K} -descriptive length of a subset V is just the minimal length needed to express it in the language \mathcal{K} .

Here is a simple example illustrating the definitions.

Example: Consider a structured set depicted in Figure 1. The universe $U = \{1, 2, 3, 4, 5\}$. The alphabet $\Sigma =$

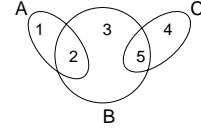


Figure 1: A structured set.

$\{A, B, C\}$. The interpretation function is $[A] = \{1, 2\}$, $[B] = \{2, 3, 5\}$, and $[C] = \{4, 5\}$. The language $\mathcal{L}(U, \Sigma)$ includes expressions like: $s_1 = (A - B) - C$, $s_2 = A - B$ and $s_3 = (B - A) - C$, with $[s_1] = [A - B] - [C] = ([A] - [B]) - [C] = \{1\} = [s_2]$ and $[s_1] = [B - A] - [C] = ([B] - [A]) - [C] = \{3\}$. The first two strings s_1 and s_2 are equivalent, but s_2 is shorter in length, therefore s_1 is strictly reducible to s_2 . It's not difficult to check that s_2 is $\mathcal{L}(U, \Sigma)$ -compact, so $\|\{1\}\| = 2$. ■

Our first algorithmic problem is: what is the complexity of determining the minimum length of a subset in the language \mathcal{K} . We pose it as a decision problem.

DEFINITION 5 (THE \mathcal{K} -MDL DECISION PROBLEM).

- **INSTANCE:** A structured set (U, Σ) , a subset $V \subseteq U$, and a positive integer $k > 0$.
- **QUESTION:** $\|V\|_{\mathcal{K}} \leq k$?

PROPOSITION 1. *The \mathcal{L} -MDL decision problem is NP-complete.*

The proof of Proposition 1 requires the simple observation that for any structured set (U, Σ) , there is a naturally induced set cover, written U/Σ , on U given by $U/\Sigma = \{[\sigma] : \sigma \in \Sigma\}$. The general minimum set-cover problem [2] easily reduces to the general \mathcal{L} -MDL problem.

The next few sections will focus on some specific structures that are relevant to realistic databases.

3. PARTITION IS IN P

In this section we focus our attention on the simple case where the symbols in Σ form a partition of U .

DEFINITION 6 (PARTITION). *A structured set (U, Σ) is a partition if the induced set cover U/Σ partitions U .*

Example: Consider these streets: Grand, Canal, Broadway in the city NewYork, VanNess, Market, Mission in SanFrancisco and Victoria, DeRivoli in Paris. The street names form the universe, which is partitioned by the alphabet consisting of the three city names, as shown in Figure 2. ■

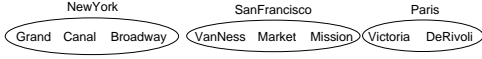


Figure 2: A partition

PROPOSITION 2. *The \mathcal{L} -MDL decision problem for a partition (U, Σ) can be solved in $\mathcal{O}(|U| \cdot \log |U|)$.*

The \mathcal{L} -MDL decision problem is particularly easy because given a subset V , $\|V\|_{\mathcal{L}}$ is simply the number of cosets that cover V exactly. Given the partition and V , computing the number of cosets that cover V exactly can be done in $\mathcal{O}(|U| \log |V|)$, and can in fact be further optimized to $\mathcal{O}(|V|)$ if special data structures are used.

Of course, in general not all subsets of street names can be expressed only by using city names – that is, the propositional language $\mathcal{L}(U, \Sigma)$ for a partition is not, in general, granular. We therefore extend the alphabet Σ to be granular; since Σ is a partition, this requires having a symbol in Σ for each element of U .

DEFINITION 7 (GRANULAR PARTITION). *A structured set (U, Σ) is a granular partition if $\Sigma = \Sigma_0 \dot{\cup} U$ where (U, Σ_0) is a partition. The interpretation function $[\cdot] : \Sigma \rightarrow \mathbf{Pwr}(U)$ is extended such that $[u] = \{u\}$ for any $u \in U$.*

The \mathcal{L} -MDL decision problem for granular partitions is also solvable in polynomial time. We first define a sub-language $\mathcal{N}_{\text{par}} \subseteq \mathcal{L}$ consisting of expressions which we refer to as *normal*, and show that all expressions in \mathcal{L} are reducible to ones in \mathcal{N}_{par} , and use this to constructively show that the \mathcal{N}_{par} -MDL decision problem is solvable in polynomial time.

Let $A = \{a_1, a_2, \dots, a_n\} \subseteq \Sigma$ be a set of symbols. We write $\vec{A} = a_1 + a_2 + \dots + a_n$. The ordering of the symbols a_i does not change the semantic evaluation nor its length, so \vec{A} can be any of the strings that are equivalent to $a_1 + a_2 + \dots + a_n$ up to the permutations of $\{a_i\}$. Furthermore, we write $[A]$ to mean $[\vec{A}]$. For a set of expressions $\{s_i\}$, $\bigoplus_i s_i$ is the expression formed by concatenating s_i by the $+$ operator.

DEFINITION 8 (NORMAL FORM FOR GRANULAR PARTITIONS). *Let $(U, \Sigma_0 \dot{\cup} U)$ be a granular partition, and its propositional language be \mathcal{L} . An expression $s \in \mathcal{L}$ is in normal form if it is of the form $(\vec{\Omega} + \vec{A}^+) - \vec{A}^-$ where $\Omega \subseteq \Sigma_0$ and A^+ and A^- are elements in U interpreted as symbols in Σ . The normal expression s is trim if $A^+ = [s] - [\Omega]$ and $A^- = [\Omega] - [s]$.*

Let $\mathcal{N}_{\text{par}}(U, \Sigma)$ be all the normal expressions in $\mathcal{L}(U, \Sigma)$ that are trim.

Intuitively, a normal form expression consists of taking the union of some set of symbols Ω from the alphabet, adding to it some elements from the universe, and subtracting some others. The expression is trim if we only add and subtract exactly those symbols that we need to express a particular subset.

Remark: Note that all normal and trim expressions $s \in \mathcal{N}_{\text{par}}$ are uniquely determined by their semantics $[s]$ and the high-level symbols Ω used. Therefore we can write $\pi(V/\Omega)$ to mean the normal and trim expression of the form $\vec{\Omega} + \vec{A}^+ - \vec{A}^-$ where $A^+ = V - [\Omega]$ and $A^- = [\Omega] - V$.

PROPOSITION 3. *A normal expression for a granular partition is \mathcal{L} -compact only if it is trim.*

LEMMA 1 (NORMALIZATION). *Every expression in \mathcal{L} is reducible to one in normal form.*

Lemma 1 immediately implies the following.

THEOREM 1. *For all $V \subseteq U$, we have $\|V\|_{\mathcal{N}_{\text{par}}} = \|V\|_{\mathcal{L}}$.*

By Theorem 1, one only needs to focus on the \mathcal{N}_{par} -MDL problem for granular partitions. The necessary and sufficient condition for \mathcal{N}_{par} -compactness can be easily stated in terms of the symbols used.

Suppose $V \subseteq U$, let us denote

$$\Sigma^+(V) = \{\sigma \in \Sigma : |\sigma \cap V| > |\sigma| - |V| + 1\},$$

and very similarly

$$\Sigma^\#(V) = \{\sigma \in \Sigma : |\sigma \cap V| \geq |\sigma| - |V| + 1\}.$$

Intuitively, the interpretation of a symbol in $\Sigma^+(V)$ includes more elements in V than elements not in V – by a difference of at least two. Similarly for a symbol in $\Sigma^\#(V)$, the difference is at least one.

We say that symbols in $\Sigma^\#(V)$ are efficient with respect to V and ones in $\Sigma^+(V)$ are strictly efficient. Symbols that are not in $\Sigma^\#(V)$ are inefficient with respect to V .

Example: Consider the partition in Figure 2. Let $V_1 = \{\text{Victoria, DeRivoli}\}$, and $V_2 = \{\text{Grand, Canal}\}$. $\Sigma^\#(V_1) = \Sigma^+(V_1) = \{\text{Paris}\}$, and $\Sigma^\#(V_2) = \{\text{NewYork}\}$, and $\Sigma^+(V_2) = \emptyset$. ■

LEMMA 2. *Let $s = (\vec{\Omega} + \vec{A}^+) - \vec{A}^-$ be an expression in \mathcal{N}_{par} representing V . It is \mathcal{N}_{par} -compact if and only if $\Sigma^+(V) \subseteq \Omega \subseteq \Sigma^\#(V)$.*

Intuitively Lemma 2 tells us that an expression is \mathcal{N}_{par} -compact if and only if it uses all strictly efficient symbols, and never uses any inefficient ones.

COROLLARY 1. *Let (U, Σ) be a granular partition. Given any $V \subseteq U$, $\pi(V/\Sigma^\#(V))$ is \mathcal{L} -compact.*

Computing $\pi(V/\Sigma^\#(V))$ is certainly in polynomial time.

THEOREM 2. *The \mathcal{L} -MDL problem for granular partitions can be solved in polynomial time.*

Example: Consider V_1 and V_2 as defined in the previous example. By Lemma 2, both of the following expressions of $V_1 \cup V_2$ are compact.

$s_1 = (\text{NewYork} + \text{Paris}) - \text{Broadway}$, and

$s_2 = \text{Paris} + (\text{Grand} + \text{Canal})$.

Note $\pi(V/\Sigma^\#(V))$ is s_1 . ■

4. HIERARCHY IS IN P

Partition has the nice property that its MDL problem is simple. However it does not adequately express many realistic structures. We shall generalize the notion of (granular) partitions to (granular) multi-level hierarchies.

DEFINITION 9 (HIERARCHY). *A structured set (U, Σ) is a hierarchy if $\Sigma = \Sigma_1 \dot{\cup} \Sigma_2 \dot{\cup} \Sigma_3 \dots \dot{\cup} \Sigma_N$, such that for any $i \leq N$, (U, Σ_i) is a partition; furthermore, for any $i, j \leq N$, we have $i < j \implies U/\Sigma_i$ refines U/Σ_j . The integer N is referred as the number of levels or the height of the hierarchy, and (U, Σ_i) the i -th level.*

Example: We extend the partition in Figure 2 to form a hierarchy with three levels ($N = 3$) shown in Figure 3. The first level has Σ_1 being the street names, the second has Σ_2 being the city names, and finally the third level has Σ_3 having only one symbol STORE. ■

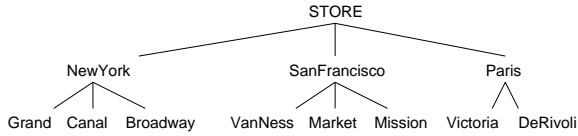


Figure 3: The STORE dimension as a tree

Consider a hierarchy $(U, \Sigma_1 \dot{\cup} \Sigma_2 \dots \dot{\cup} \Sigma_N)$. First note that it is granular if and only if in the first level $\Sigma_1 = U$, i.e. $(U, \Sigma_1 \dot{\cup} \Sigma_2)$ is a granular partition. For $i < N$, we define $\Delta_i = \bigcup_{k=i+1}^N \Sigma_k$. The alphabet Δ_i contains all symbols in levels higher than the i -th level of the hierarchy. We may view Σ_i as a universe, and consider (Σ_i, Δ_i) as a new hierarchy, with the interpretation function given by,

$$[\cdot]_i : \Delta_i \rightarrow \mathbf{Pwr}(\Sigma_i) : \lambda \mapsto \{\sigma \in \Sigma_i : [\sigma] \subseteq [\lambda]\}.$$

Let \mathcal{L}_i denote the propositional language $\mathcal{L}(\Sigma_i, \Delta_i)$.

Much of the discussion regarding partitions naturally applies to hierarchies with some appropriate generalization.

DEFINITION 10 (NORMAL FORMS). *An expression $s \in \mathcal{L}_i$ is in normal form for the hierarchy if it is of the form $s = \hat{s} + \overrightarrow{\Omega_i^+} - \overrightarrow{\Omega_i^-}$, where $\hat{s} \in \mathcal{L}_{i+1}$ is the leading sub-expression of s , and $\Omega_i^+, \Omega_i^- \subseteq \Sigma_i$.*

It is trim if \hat{s} is \mathcal{L}_{i+1} -compact and $\Omega_i^+ = [s]_i - [\hat{s}]_i$ and $\Omega_i^- = [\hat{s}]_i - [s]_i$.

We denote $(\mathcal{N}_{\text{hie}})_i = \mathcal{N}_{\text{hie}}(\Sigma, \Delta_i)$ to be the set of all normal and trim expressions of the hierarchy (Σ_i, Δ_i) , and let $\mathcal{N}_{\text{hie}} \equiv (\mathcal{N}_{\text{hie}})_1$.

Here are some familiar results.

PROPOSITION 4. *A normal expression in \mathcal{L}_i is \mathcal{L}_i -compact only if it is trim.*

LEMMA 3 (NORMALIZATION). *Every expression in \mathcal{L}_i can be reduced to one in $(\mathcal{N}_{\text{hie}})_i$.*

THEOREM 3. *Let (U, Σ) be a hierarchy, then for any $V \subseteq U$, $\|V\|_{\mathcal{L}} = \|V\|_{\mathcal{N}_{\text{hie}}}$.*

Theorem 3 follows immediately from Lemma 3.

As in the case for partitions, one only needs to focus on the expressions in \mathcal{N}_{hie} since \mathcal{N}_{hie} -compactness implies \mathcal{L} -compactness.

LEMMA 4 (NECESSARY CONDITION). *Let $s \in (\mathcal{N}_{\text{hie}})_i$, and $V = [s]$. It is $(\mathcal{N}_{\text{hie}})_i$ -compact only if $\Sigma_{i+1}^+(V) \subseteq [\hat{s}]_{i+1} \subseteq \Sigma_{i+1}^\#(V)$, where $\Sigma_{i+1}^+(V)$ and $\Sigma_{i+1}^\#(V)$ are respectively the strictly efficient and efficient alphabets in Σ_{i+1} with respect to V .*

Note that Lemma 4 mirrors Lemma 2. It states that the expression s is compact only when \hat{s} expresses all the efficient symbols in Σ_{i+1} with respect to V , and never any inefficient ones. It is also worth noting that this condition is not sufficient, unlike the case in Lemma 2.

For any $i \leq N$, define a partial order \preceq over $(\mathcal{N}_{\text{hie}})_i$, such that for any two expressions $s, t \in (\mathcal{N}_{\text{hie}})_i$,

$$s \preceq t \iff [s]_i = [t]_i \text{ and } [\hat{s}]_{i+1} \supseteq [\hat{t}]_{i+1}.$$

PROPOSITION 5. *Let s, t be two equivalent expressions in \mathcal{N}_{hie} who satisfy the necessary condition of Lemma 4. Then $s \preceq t \implies \|s\| \leq \|t\|$. In other words, $\|\cdot\| : (\mathcal{N}_{\text{hie}}, \preceq) \rightarrow (\mathbb{N}, \leq)$ is order preserving.*

Therefore by minimizing with respect to \preceq , we are effectively minimizing the length. It is immediate from the definition of \preceq that minimization over \preceq in $(\mathcal{N}_{\text{hie}})_i$ yields maximization of $[\hat{s}]_{i+1}$ which is bounded by $\Sigma_{i+1}^\#([s])$.

We are now ready to present a decomposition procedure to compute minimal expressions in \mathcal{N}_{hie} for a given subset $V \subseteq U$.

DEFINITION 11 (DECOMPOSITION OPERATORS). *Define the following mappings for each $i \leq N$:*

- $\Phi_i : \mathbf{Pwr}(\Sigma_i) \rightarrow \mathbf{Pwr}(\Sigma_{i+1}) : V \mapsto \Sigma_{i+1}^\#(V)$.
- $\Psi_i^+ : \mathbf{Pwr}(\Sigma_i) \rightarrow \mathbf{Pwr}(\Sigma_i) : V \mapsto V - [\Phi_i(V)]_i$, and
- $\Psi_i^- : \mathbf{Pwr}(\Sigma_i) \rightarrow \mathbf{Pwr}(\Sigma_i) : V \mapsto [\Phi_i(V)]_i - V$.

THEOREM 4. *Suppose $V \subseteq U$. Let,*

- $V_1 = V$,
- $V_{i+1} = \Phi_i(V_i)$, $W_i^+ = \Psi_i^+(V_i)$ and $W_i^- = \Psi_i^-(V_i)$, for $1 < i \leq N$.

Define the expressions,

- $s_N = \overrightarrow{V_N}$,
- $s_{i-1} = \left(s_i + \overrightarrow{W_{i-1}^+} \right) - \overrightarrow{W_{i-1}^-}$ for $1 \leq i < N$.

Then s_1 is a \mathcal{N}_{hie} -compact expression expressing V .

Clearly the complexity of construction of s_1 is in polynomial time, in fact can be done in $\mathcal{O}(|\Sigma| \cdot |V| \cdot \log |V|)$. The algorithm is paraphrased in Figure 4.

Example: Consider the hierarchy in Figure 3. Let $V_1 = \{\text{Victoria, DeRivoli, Grand, Broadway, Market}\}$. The algorithm produces:

$V_2 = \{\text{Paris, NewYork}\}$, and $W_1^+ = \{\text{Market}\}$, $W_1^- = \{\text{Canal}\}$.
 $V_3 = \{\text{STORE}\}$, $W_2^+ = \emptyset$, $W_2^- = \{\text{SanFrancisco}\}$.

The expressions produced by the algorithm are:

$s_3 = \text{STORE}$.

$s_2 = \text{STORE-SanFrancisco}$.

$s_1 = (\text{STORE-SanFrancisco}) + \text{Market-Canal}$.

Since s_1 is guaranteed compact, $\|V_1\| = \|s_1\| = 4$. Note s_1 is not the only compact expression, $(\text{NewYork-Canal}) + \text{Market} + \text{Paris}$ for instance is another expression with length 4. ■

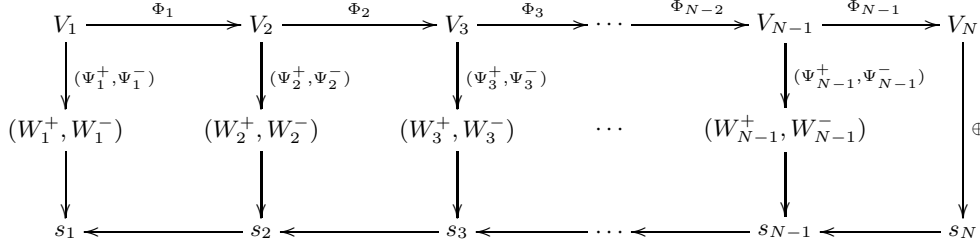


Figure 4: The decomposition algorithm for hierarchy

5. MULTIDIMENSIONAL PARTITION AND HIERARCHY ARE NP-COMPLETE

An important family of structures is the multidimensional structures. The simplest is the multidimensional partition.

DEFINITION 12 (MULTIDIMENSIONAL PARTITION). A structure (U, Σ) is a multidimensional partition if the alphabet $\Sigma = \Sigma_1 \dot{\cup} \Sigma_2 \cdots \dot{\cup} \Sigma_N$ where for every i , (U, Σ_i) is a partition as defined in Definition 6. The integer N is the dimensionality of the structure. The hierarchy (U, Σ_i) is the i -th dimension.

Note the subtle difference between a multidimensional partition and a hierarchy. A hierarchy has the additional constraint that U/Σ_i are ordered, and is in fact a special case of the multidimensional partition, but as one might expect and we shall show that the relaxed definition of multidimensional partition leads to a NP-hard MDL-problem.

A simple extension of the multidimensional partition is the multidimensional hierarchy.

DEFINITION 13 (MULTIDIMENSIONAL HIERARCHY). A structure (U, Σ) is a multidimensional hierarchy if the alphabet $\Sigma = \Sigma_1 \dot{\cup} \Sigma_2 \cdots \dot{\cup} \Sigma_N$ where for every i , (U, Σ_i) is a hierarchy as defined in Definition 9. The integer N is the dimensionality of the structure.

In this section, we will consider three languages which express subsets of the universe, with successively more grammatical freedom. It will be shown that the MDL decision problem is NP-complete for all three languages. In fact, we will show this on a specific kind of structures Σ that we call *product structures*. Intuitively, multidimensional partitions and multidimensional hierarchies make sense when the elements of the universe can be thought of as N -dimensional points, and each of the partitions or hierarchies operates along one dimension. Most of our discussion will focus on the 2-dimensional case ($N = 2$), which is enough to yield the NP-completeness results. We next define product structures for the 2D case.

DEFINITION 14 (2-D PRODUCT STRUCTURE). We say that (U, Σ) is a 2-D product structure if universe U is the cartesian product of two disjoint sets X and Y : $U = X \times Y$, and the alphabet Σ is the union of X and Y : $\Sigma = X \dot{\cup} Y$. The interpretation function is defined as, for any $z \in \Sigma$,

$$[z] = \begin{cases} \{z\} \times Y & \text{if } z \in X, \\ X \times \{z\} & \text{if } z \in Y. \end{cases}$$

Note that the 2-D product structure is granular, since the language $\mathcal{L}(X \times Y, \Sigma)$ can express every singleton $\{(x, y)\} \in \mathbf{Pwr}(U)$ by the expression $(x \cdot y)$.

The 2-D product structure admits two natural expression languages, both requiring the notion of product expressions.

DEFINITION 15 (PRODUCT EXPRESSIONS). An expression $s \in \mathcal{L}$ is a product expression if it is of the form $s = (\vec{A} \cdot \vec{B})$ where $A \subseteq X$ and $B \subseteq Y$.

We build up two languages using product expressions.

DEFINITION 16 (DISJUNCTIVE PRODUCT LANGUAGE). The disjunctive product language \mathcal{L}_P^+ is defined as,

- $\epsilon \in \mathcal{L}_P^+$,
- any product expression s belongs to \mathcal{L}_P^+ ,
- if $s, t \in \mathcal{L}_P^+$, then $(s + t) \in \mathcal{L}_P^+$.

It is immediate that any expression $s \in \mathcal{L}_P^+$ can be written in the form $\bigoplus_{i \in I} s_i$ where for any i , s_i is a product expression.

A generalization of the disjunctive product language is to allow other operators to connect the product expressions.

DEFINITION 17 (PROPOSITIONAL PRODUCT LANGUAGE). The propositional product language \mathcal{L}_P is defined as,

- $\epsilon \in \mathcal{L}_P$,
- any product expression s belongs to \mathcal{L}_P ,
- if $s, t \in \mathcal{L}_P^+$, then $(s + t), (s - t), (s \cdot t) \in \mathcal{L}_P$.

Obviously $\mathcal{L}_P^+ \subsetneq \mathcal{L}_P \subsetneq \mathcal{L}$.

Example:

Consider a 2-D product structure with $\text{CITY} = \{\text{New York, San Francisco, Paris}\}$, and $\text{PRODUCT} = \{\text{Clothing, Beverage, Automobile}\}$. The universe $U = \text{CITY} \cdot \text{PRODUCT}$ consists of the 9 pairs of city

NewYork,	Clothing
NewYork,	Beverage
NewYork,	Automobile
SanFrancisco,	Clothing
SanFrancisco,	Beverage
SanFrancisco,	Automobile
Paris,	Clothing
Paris,	Beverage
Paris,	Automobile

name and product family: $U =$

The alphabet Σ consists of 6 symbols $\Sigma = \text{CITY} \dot{\cup} \text{PRODUCT} = \{\text{NewYork, SanFrancisco, Paris, Clothing, Beverage, Automobile}\}$.

The interpretation of a symbol are the pairs in U which the symbol occurs in.

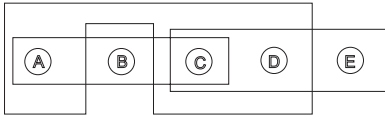


Figure 5: A set cover with three cosets.

$$\text{For instance, [Beverage]} = \left\{ \begin{array}{l} (\text{NewYork}, \text{Beverage}) \\ (\text{SanFrancisco}, \text{Beverage}) \\ (\text{Paris}, \text{Beverage}) \end{array} \right\}.$$

Consider the following expressions in $\mathcal{L}(U, \Sigma)$:

$$\begin{aligned} s_1 &= ((\text{NewYork}+\text{Paris}) \cdot \text{Clothing}) + (\text{NewYork} \cdot \text{Beverage}), \\ s_2 &= (\text{NewYork}+\text{Paris} \cdot (\text{Clothing}+\text{Beverage})) - (\text{NewYork} \cdot \text{Clothing}), \\ s_3 &= \text{NewYork} \cdot \text{Beverage}. \end{aligned}$$

The expression $s_1 \in \mathcal{L}_P^+$, $s_2 \in \mathcal{L}_P - \mathcal{L}_P^+$, and $s_3 \in \mathcal{L} - \mathcal{L}_P$. They are evaluated to

$$[s_1] = \{(\text{NewYork}, \text{Clothing}), (\text{Paris}, \text{Clothing}), (\text{NewYork}, \text{Beverage})\},$$

and $[s_2] = \left\{ \begin{array}{l} (\text{NewYork}, \text{Beverage}), (\text{NewYork}, \text{Automobile}), \\ (\text{Paris}, \text{Clothing}), (\text{Paris}, \text{Beverage}) \end{array} \right\}.$

The last expression s_3 is a bit tricky – it contains all tuples of NewYork that are not Beverage, so

$$[s_3] = \{(\text{NewYork}, \text{Clothing}), (\text{NewYork}, \text{Automobile})\}. \blacksquare$$

we will see that the MDL decision problem for each of these languages is NP-complete.

5.1 Complexity of the 2-D MDL decision problems

In this section, we prove that each of the three languages has a NP-hard MDL decision problem. It's obvious that they are all in NP.

The proof is by a reduction from the 3-eXact Set Cover (3XSC) problem.

Recall that an instance of 3XSC problem consists of a set cover $\mathcal{C} = \{C_1, C_2, \dots, C_n\}$ where $(\forall C \in \mathcal{C})|C| = 3$ and an integer $k > 0$. The question is if there exists a sub-cover $\mathcal{D} \subseteq \mathcal{C}$ such that $\bigcup \mathcal{D} = \bigcup \mathcal{C}$ and $|\mathcal{D}| \leq k$. This is known to be NP-complete [2].

From this point on, we fix the instance of the 3XSC (\mathcal{C}, k) . Write $\mathcal{C} = \{C_1, C_2, \dots, C_n\}$. Define $X = \bigcup \mathcal{C}$, and for each $i \leq n$, let Y_i be a set such that $|Y_i| = m > 3$. The family $\{Y_i\}_n$ is made disjoint. Let $Y = \left(\bigcup_{i \leq n} Y_i\right) \dot{\cup} \{y_*\}$, where y_* does not belong to any Y_i . The structure is the 2-D product structure of $X \times Y$. The subset to be represented is given by $V = \bigcup_{i \leq n} (C_i \times Y_i) \cup (X \times \{y_*\})$. It is not difficult to see that this is a polynomial time reduction.

Example: Consider a set $X = \{A, B, C, D, E\}$, and a cover $\mathcal{C} = \{C_1, C_2, C_3\}$ where $C_1 = \{A, B, C\}$, $C_2 = \{C, D, E\}$ and $C_3 = \{A, C, D\}$.

It is transformed by first constructing Y_1, Y_2 and Y_3 , all disjoint and each with 4 elements.

Then let $Y = Y_1 \dot{\cup} Y_2 \dot{\cup} Y_3 \dot{\cup} \{y_*\}$. The structure is the 2-D product structure of X and Y . The subset $V = (C_1 \times Y_1) \dot{\cup} (C_2 \times Y_2) \dot{\cup} (C_3 \times Y_3) \dot{\cup} (X \times \{y_*\})$. It is shown as the shaded boxes in Figure 6. \blacksquare

It turns out that for this very specific subset V , one can characterize the form of the compact expressions that express V .

PROPOSITION 6. All \mathcal{L}_P -compact expressions of V are in the form of

$$s = \bigoplus_{i \in I} (\vec{C}_i \cdot \vec{Y}_i) + \bigoplus_{j \in J} (\vec{C}_j \cdot \vec{Y}_j^*),$$

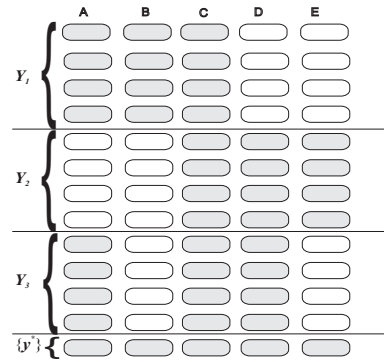


Figure 6: The transformed instance of the MDL problem of 2-D product structure.

where $Y_j^* = Y_j \dot{\cup} \{y_*\}$, and $I \cap J = \emptyset$, and $I \dot{\cup} J = \{1, 2, \dots, n\}$.

Note that, by Proposition 6, the \mathcal{L}_P -compact expressions of V do not make use of the negation “-” and conjunction “.” operators between product expressions, hence they belong to \mathcal{L}_P^+ .

Example: For subset V in Figure 6, the expression

$$s = (\vec{C}_1 \cdot \vec{Y}_1^*) + (\vec{C}_2 \cdot \vec{Y}_2) + (\vec{C}_3 \cdot \vec{Y}_3^*)$$

is both \mathcal{L}_P and \mathcal{L}_P^+ -compact. Therefore $\|V\|_{\mathcal{L}_P} = \|V\|_{\mathcal{L}_P^+} = (3 + 5) + (3 + 4) + (3 + 5) = 23$. \blacksquare

THEOREM 5. \mathcal{L}_P^+ -MDL and \mathcal{L}_P -MDL's are NP-complete.

PROOF. This follows from Proposition 6. As we mentioned, $\|V\|_{\mathcal{L}_P^+} = \|V\|_{\mathcal{L}_P}$. Let s be a \mathcal{L}_P -compact expression of V . Since

$$s = \bigoplus_{i \in I} (\vec{C}_i \cdot \vec{Y}_i) + \bigoplus_{j \in J} (\vec{C}_j \cdot \vec{Y}_j^*),$$

its length is $\|s\| = \sum_{i \leq n} |C_i| + |Y_i| + |J| = (3 + m)n + |J|$. Since $[s] = V$, it is necessarily the case that $X \times \{y_*\} \subseteq \bigoplus_{j \in J} (\vec{C}_j \cdot \vec{Y}_j^*)$, or that $\{C_j\}_{j \in J}$ covers X .

Minimizing $\|s\|$ with s in the given form is equivalent to minimization of $|J|$, or finding a minimal cover of X which is of course the objective of the 3-XSC problem. \square

As for \mathcal{L} , Proposition 6 does not hold.

Example: Consider once again the subset V in Figure 6, and the expression in \mathcal{L} but not in \mathcal{L}_P :

$$s = (\vec{A} - \vec{Y}_2) + (\vec{B} \cdot \vec{Y}_1^*) + C + (\vec{D} - \vec{Y}_1) + (\vec{E} \cdot \vec{Y}_2^*).$$

Note that $[s] = V$, but certainly s is not of the form of Proposition 6. Its length is $\|s\| = (1 + 4) + (1 + 5) + 1 + (1 + 3) + (1 + 4) = 21$. Therefore in this case we have that $\|V\|_{\mathcal{L}} < \|V\|_{\mathcal{L}_P}$. \blacksquare

The richness of \mathcal{L} prevents us from using Proposition 6 to arrive at the NP-hardness of the \mathcal{L} -MDL decision problem. We have to modify the reduction for the 3-XSC problem, and deal with the expressions in greater detail.

DEFINITION 18 (DOMAIN DEPENDENCY). Let $X_0 = X$ and $Y_0 = Y$ as defined in the reduction from a 3-XSC problem.

Define a sequence of sets X_0, X_1, X_2, \dots , and Y_0, Y_1, Y_2, \dots , such that for all $k \geq 0$, $X_{k+1} = X_k \dot{\cup} \{\alpha_k\}$ and $Y_{k+1} = Y_k \dot{\cup} \{\beta_k\}$, where α_k and β_k are two symbols that do not belong to X_k and Y_k respectively.

We therefore have a family of 2-D product structures $\{X_k \times Y_k\}$ with the propositional languages $\mathcal{L}_0 \subsetneq \mathcal{L}_1 \subsetneq \mathcal{L}_2 \dots$.

Let $s \in \mathcal{L}_k$, for $k' \geq k$, write $[s]_{k'}$ to be the evaluation of the expression s in the language $\mathcal{L}_{k'}$.

For any $k \geq 0$, we say that $s \in \mathcal{L}_k$ is domain independent if

$$\forall k' > k. [s]_{k'} = [s]_k.$$

If $s \in \mathcal{L}_k$ is not domain independent, then it's domain dependent.

The notion of domain dependency naturally bi-partitions the languages. Let $\mathcal{L}_k^I = \{s \in \mathcal{L}_k : s \text{ is domain independent}\}$, and $\mathcal{L}_k^D = \{s \in \mathcal{L}_k : s \text{ is domain dependent}\}$.

Given an expression s , whether it is domain dependent or not depends on set of unbounded symbols, defined below.

DEFINITION 19 (BOUNDED EXPRESSIONS). Let s be an expression in a propositional language. The set of unbounded symbols of s , $\mathbb{U}(s)$ is a set of symbols that appear in s , defined as: $\mathbb{U}(\epsilon) = \emptyset$, $\mathbb{U}(\sigma) = \{\sigma\}$, and $\mathbb{U}(t + t') = \mathbb{U}(t) \cup \mathbb{U}(t')$, $\mathbb{U}(t - t') = \mathbb{U}(t) - \mathbb{U}(t')$, $\mathbb{U}(t \cdot t') = \mathbb{U}(t) \cap \mathbb{U}(t')$.

In case $\mathbb{U}(s) = \emptyset$, we say that s is a bounded expression, or that it is bounded, otherwise s is unbounded.

PROPOSITION 7. An expression is domain independent if and only if it is bounded, i.e.

$$\forall k. s \in \mathcal{L}_k^I \iff \mathbb{U}(s) = \emptyset.$$

The importance of domain dependency of expressions is demonstrated by the following results.

PROPOSITION 8. Let $V \subseteq X_0 \times Y_0$. We have

$$\begin{aligned} \forall k \geq 0. \|V\|_{\mathcal{L}_k^I} &\geq \|V\|_{\mathcal{L}_{k+1}^I}, \text{ and} \\ \forall k \geq 0. \|V\|_{\mathcal{L}_k^D} &\leq \|V\|_{\mathcal{L}_{k+1}^D}. \end{aligned}$$

COROLLARY 2. For any $V \subseteq X_0 \times Y_0$,

$$\forall k > (2|V|). \|V\|_{\mathcal{L}_k^I} \leq \|V\|_{\mathcal{L}_k^D}.$$

In other words, $\forall k > (2|V|). \|V\|_{\mathcal{L}_k} = \|V\|_{\mathcal{L}_k^I}$.

Therefore by enlarging the dimensions X and Y by adding $2|V|$ new symbols to each, we are guaranteed that all \mathcal{L} -compact expressions are domain independent, and hence are bounded.

Note that every expression in \mathcal{L}_P is bounded, so $\mathcal{L}_P \subseteq \mathcal{L}^I$. In fact we have the following strong result.

PROPOSITION 9. For the V resulting from our reduction from an 3-XSC problem, $\|V\|_{\mathcal{L}_P} = \|V\|_{\mathcal{L}^I}$.

Remark: It is worth noting that there are \mathcal{L}^I -compact expressions that are not in the form given in Proposition 6.

THEOREM 6. The \mathcal{L} -MDL decision problem is NP-complete.

Since the 2-D product structure is a specific case of the multidimensional partition and the multidimensional hierarchy, the lower bound applies also to these structures:

COROLLARY 3. The \mathcal{L} -MDL decision problems for the general multidimensional partitions and multidimensional hierarchies are NP-complete.

5.2 Related work

The \mathcal{L}_P^+ -MDL (and the \mathcal{L}_P -MDL) decision problem for 2-D product structure closely resembles the rectangle covering problem of 2-D axis-aligned polygons (with holes), a well-known problem [2] that has found application in computer vision, computer aided design and recently in spatial database ([7], [1]), and has been well studied in computational geometry ([6], [8], [9]).

However, our problem differs from this in two important ways. First, we deal with discrete, categorical, unordered data, instead of linearly ordered dense dimensions. Second, the measure to be minimized in the work above is the number of rectangles, while our objective is to minimize the length of the the expression. These two measures do not always agree.

Example: Consider the 2-D product structure formed by $X = \{1, 2, 3, 4, 5\}$ and $Y = \{a, b\}$. The subset V to be represented is shown in Figure 7. The solid line shows the

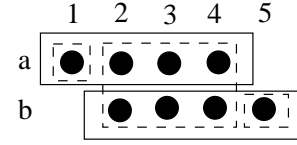


Figure 7: A subset in a 2-D product structure.

rectangles used to cover V , in this case a minimum of two is needed. The corresponding expression in \mathcal{L}_P is $(1 + 2 + 3 + 4) \cdot a + (2 + 3 + 4 + 5) \cdot b$; its length is 10. But consider covering V with three rectangles shown in the dashed lines. The expression in \mathcal{L}_P for the three rectangles is $(1 \cdot a) + (2 + 3 + 4) \cdot (a + b) + (5 \cdot b)$, and its length is 9. In this case, by using more rectangles, we end up using less symbols. ■

In [1] and [7], the MDL-principle was used to summarize answer sets resulting from mining of multidimensional data. These answer sets are subsets of product structures. These works treat them as multidimensional polygons, and use heuristics to try to cover the polygons by rectangles. The “description length” is defined to be the number of rectangles used; this problem is NP-complete, as it falls into the class of polygon covering problems. As mentioned above, we treat dimensions as unordered and use expression length rather than number of rectangles as the measure to optimize.

Lakshmanan et. al. in [7] proposed an interesting algorithm in polynomial time for computing optimal expressions for hierarchical categorical structures. This can be done because their language is a subset of ours; in particular, it does not allow product expressions, the feature that causes the NP-hardness in our case.

5.3 Heuristics

It is natural to consider heuristic solutions. Despite the differences noted above between the polygon covering problem and the multidimensional MDL problem, there are certainly strong similarities in the flavour of the formulation; hence a good starting point for developing heuristics for our problem would be to deploy some existing heuristics found in the literature.

Due to the vast interest in the covering problem, a number of well studied heuristics are known. These heuristics have very different properties. The ones from the database community often are more scalable $\mathcal{O}(n^2)$ [1], but have no

performance guarantees, and the ones from the computational geometry community can be very accurate—the best known approximation bound on the heuristics of the rectangle cover problem is $\mathcal{O}(\sqrt{\log n})$ [6]—but can be very expensive, e.g. $\mathcal{O}(n^6)$ [3].

Utilizing the existing algorithms is fairly straightforward. We outline the steps as follows for the 2-D case for simplicity; it can be readily generalized to higher dimensions.

1. Order X and Y arbitrarily, and treat the elements in the universe $X \times Y$ as 2-D points.
2. Declare elements in V to be the interior points of an axis aligned polynomial P .
3. Apply a heuristic to find the minimal rectangular cover $\{R_1, R_2, \dots, R_K\}$ of the polynomial P .
4. Convert each rectangle R_k to a product expression $t_k = (\overrightarrow{A_k} \cdot \overrightarrow{B_k})$ where $A_k \subseteq X$ and $B_k \subseteq Y$.
5. If any two product expressions t_i and t_j agree on a dimension, i.e. $A_i = A_j$ or $B_i = B_j$, then merge them into one: if $A_i = A_j$, then merge them into $(\overrightarrow{A_i} \cdot \overrightarrow{B_i \cup B_j})$; similarly, if $B_i = B_j$, then form $(\overrightarrow{A_i \cup A_j} \cdot \overrightarrow{B_i})$.
6. Form the final expression by concatenating the remaining product expressions by disjunction (+).

The generalization to higher dimensions requires the heuristics in step 3 to be able to handle higher dimension polygons. Most of the accurate heuristics whose approximation bounds are known are specifically for 2D polygons, but the maximal growth algorithm found in [1] can certainly be used in high dimensions. The merge rule in step 5 must check that t_i and t_j agreeing on all but at most one dimension.

By no means does this preserve the approximation accuracy of the polygon cover heuristic.

6. APPLICATIONS OF COMPACT EXPRESSIONS

First, as we mentioned above, MDL has been proposed as a guiding principle for summarization of large query results ([7], [1]). In fact in [7] the authors argue that in data mining applications it may be worthwhile to present to the user a less accurate answer set at the benefit of extra brevity. Our theory of compact expressions clearly can be directly applied to concise summarization of multidimensional query results.

Going beyond this application, we outline the use of our results for OLAP query optimization and for efficient retrieval from XML documents.

6.1 Single and multi-OLAP query optimization

In this subsection, we demonstrate that the MDL-principle can be applied to OLAP query optimization. We show that this problem is effectively an \mathcal{L} -MDL problem for a multidimensional structure. We keep the discussion informal, and explain the ideas by an on-going example.

Let us formalize an OLAP data cube with N dimensions as consisting of an N -dimensional hierarchy. The universe is the cartesian product of N sets D_1, D_2, \dots, D_N , with each

D_i representing the leaf elements of dimension i . Recall that a hierarchy is equipped with N alphabets $\Sigma_1, \Sigma_2, \dots, \Sigma_N$, each representing a hierarchy for the corresponding dimension. As always, denote $U = \prod_{i=1}^N D_i$ to be the universe.

A data cube also contains measures: one tuple of numbers for each element in U . Denote by \mathcal{M} the set of all possible measure values; for example, if there are k real-valued measures, then $\mathcal{M} = \mathbb{R}^k$. Formally, a data cube is then a function $\mathbb{D} : (U, \Sigma) \rightarrow \mathcal{M}$ mapping N -tuples of categorical leaf elements to k -tuples of (usually numeric) values. This function can be partial, corresponding to sparse data cubes, or it can be made total by augmenting the range \mathcal{M} by a unique element NULL such that $\mathbb{D}(\vec{v}) = \text{NULL}$ whenever the value for the tuple \vec{v} is undefined.

Example: Consider a two-dimensional hierarchy. The first dimension is a hierarchy with the alphabet **PRODUCT** = **PRODUCT**₁ ∪ **PRODUCT**₂ with **PRODUCT**₁ containing **Jacket**, **Sweater**, **Suite**, **Shirt**, **Soda**, **Beer**, **Spirits**, **Sedan**, **SportsCar**, **Motorcycle**; and **PRODUCT**₂ containing **Clothing**, **Beverage**, **Automobile**.

The second dimension is a hierarchy with the alphabet **STORE** = **STORE**₁ ∪ **STORE**₂ as shown in Figure 3.

The two hierarchies are shown in Figure 8 in the form of dimension tables.

PRODUCT		STORE	
Clothing	Jacket	NewYork	Canal
	Sweater		Grand
	Suite	SanFrancisco	Broadway
	Shirt		VanNess
Beverage	Soda	Paris	Market
	Beer		Mission
	Spirits		Victoria
Automobile	Sedan		DeRivoli
	MiniVan		
	MotorCycle		

Figure 8: The OLAP dimensions

A data cube \mathbb{D} formed by these two dimensions with two measures is a function mapping the pairs of product name and street name to pairs of numbers, and intuitively a two dimensional matrix with entries being pairs of numbers. The measures may be Total Sales and Sales Count, i.e. $\mathcal{M} = \mathbb{R} \times \mathbb{N}$, in which case, the mapping is $\mathbb{D} : \text{PRODUCT}_1 \times \text{STORE}_1 \rightarrow \mathbb{R} \times \mathbb{N}$. ■

By a query $q(V)$ on a data cube \mathbb{D} , we simply mean a subset V of the universe U ; V is called the range of the query. The answer to a query is the function $\mathbb{D}|V$ — the function \mathbb{D} restricted to the sub-domain V .

The language \mathcal{L} (or \mathcal{L}_P^+ , \mathcal{L}_P) can be used to describe the range of the query. Let $s \in \mathcal{L}$ be an expression, and $q(s)$ be the query with the range $[s]$. When a string s in \mathcal{L} is used in expressing the query range, we refer to $\|s\|$ as the query length. The answer to the query is written as $[q(s)] = \mathbb{D}[s]$.

Example: Consider the OLAP data cube \mathbb{D} of the previous example.

Let $V = \{(\text{Grand}, \text{Jacket}), (\text{Canal}, \text{Jacket})\}$, the query $q(V)$ asks for the Sales and SalesCount for the product **Jacket** on the streets of **Grand** and **Canal** (in **NewYork**).

The answer of the query is $\mathbb{D}|V$, which is simply:

\vec{v}		$\mathbb{D}(\vec{v})$	
Grand	Jacket	-	-
Canal	Jacket	-	-

The query $q(V)$ can also be equivalently expressed in the following forms:

$$q(s_1) = q(\text{Grand, Canal}) \cdot \text{Jacket} \text{ and } \\ q(s_2) = q(\text{NewYork-Broadway}) \cdot \text{Jacket}.$$

Both $q(s_1)$ and $q(s_2)$ produce the same answer as $q(V)$ because $[s_1] = [s_2] = V$. ■

Single OLAP Query Optimization: Given a query q with its range specified by a subset V , we want to find a compact expression s for V such that query $q(s)$ requires minimal description length but produces the same answer.

A special case of single OLAP query optimization that is especially important and motivating is when the range V is a cartesian product.

Let the universe U be $D_1 \times D_2 \times \dots \times D_N$ where each dimension D_i is in turn a hierarchy. Normally solving for the $\mathcal{L}_P(U, \Sigma)$ -compact expression of an arbitrary range V is NP-hard, but in the special case that $V = A_1 \times A_2 \times \dots \times A_N$, the problem can be solved in polynomial time.

Recall that $\mathcal{L}(D_i, \Sigma_i)$ is the propositional language of the hierarchy (D_i, Σ_i) . Let s_i be a $\mathcal{L}(D_i, \Sigma_i)$ -compact expression for A_i ; it can be found using Theorem 4. The $\mathcal{L}_P(U, \Sigma)$ -compact expression of V is $s_1 \cdot s_2 \cdot \dots \cdot s_n$.

In this case, one can argue that there is a distinct advantage to use a compact expression instead of the naive specification of the range in the context of Relational OLAP (ROLAP) applications.

We illustrate the performance gain by an example.

Example: Consider the OLAP cube \mathbb{D} in the previous example. In order to store it in a relational database, it is mapped to a star schema with two dimension tables PRODUCT and STORE, one for each of the dimensions as shown in Figure 8, and a fact table FACT as shown in Figure 9.

Name	Street	Sales	SalesCount
Jacket	Grand	\$98.34	2
Sedan	Market	\$23,034	1
⋮	⋮	⋮	⋮

Figure 9: A fact table.

The FACT's Name column joins with PRODUCT's Name column, and the Street column in FACT joins with the Street column in STORE. The star schema is shown in Figure 10.

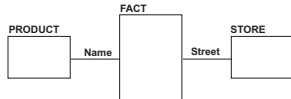


Figure 10: The star schema for an OLAP cube

It is common practice to join all three tables and create a view spanning all three tables. Assume that the view OLAPView has been created by joining the dimension tables PRODUCT, STORE with the fact table Fact.

Suppose a query has the range

$$V = \left\{ \begin{array}{ll} (\text{Jacket, Victoria}) & (\text{Jacket, DeRivoli}) \\ (\text{Sweater, Victoria}) & (\text{Sweater, DeRivoli}) \\ (\text{Suite, Victoria}) & (\text{Suite, DeRivoli}) \end{array} \right\} \\ = \{\text{Jacket, Sweater, Suite}\} \times \{\text{Victoria, DeRivoli}\}.$$

A naive translation of $q(V)$ into the SQL would result in a needlessly long statement:

```

SELECT * FROM OLAPView
WHERE
  Name IN ('Jacket', 'Sweater', 'Suite') AND
  Street IN ('Victoria', 'DeRivoli');
  
```

The \mathcal{L}_P -expression of V is $s = (\text{Clothing-Shirt}) \cdot \text{Paris}$.

It is not difficult to translate $q(s)$ into a relational SQL query:

```

SELECT * FROM OLAPView
WHERE
  Family='Clothing' AND Name <> 'Shirt' AND
  City='Paris';
  
```

Note that $q(s)$ is not only shorter than $q(V)$ as a result of the MDL-principle, but for each dimension, it also makes use of high level symbols (Paris and Clothing) instead of the low level symbols (Victoria, DeRivoli and Jacket, Sweater, Suite). The resulting SQL statement therefore has the predicate applied to higher levels in the dimensions (Family='Clothing' and City='Paris'). This yields a significant opportunity for performance gain. Since there are fewer symbols in the City level, searching for rows with City='Paris' should be faster than

Street in ('Victoria', 'DeRivoli') if the indexes are built properly on the view OLAPView. Similar argument applies to Family='Clothing' AND Name <> 'Shirt' versus Name IN ('Jacket', 'Sweater', 'Suite').

Multi-OLAP Query Optimization: Given a family of queries $\{q_i\}$ with ranges $\{V_i\}$ respectively, we wish to find a shortest query $q(s)$ producing the answer $\mathbb{D}(\bigcup_i V_i)$.

The motivation for multi-OLAP query is the following identity: $\|\bigcup_i V_i\| \leq \sum_i \|V_i\|$. A shortest expression for the union of all the ranges is always shorter or at most equal to the combined length of all the expressions of the individual ranges.

We propose that when the workload is high, it is advantageous to simultaneously consider multiple OLAP queries $\{q(V_i)\}$, and find a compact expression s for the union of the ranges $\bigcup_i V_i$. The SQL statement resulting from s is more likely to have better performance for the same reasons as in the case of optimization of a single OLAP query.

The answer set of $q(s)$ is sufficient to answer each query $q(V_i)$. Therefore in order to distribute the answers to each individual query, one simply issues $q(V_i)$ against the answer set of $q(s)$.

In realistic scenarios, the size of $q(s)$ is much smaller than the size of the fact table, so the process of distribution is not costly.

The benefit of multi-OLAP query optimization is in exploiting the structure similarity of the query ranges, and removal of redundancy among them. The same motivation prompted much work in simultaneous query optimization in [10] and [4], to name a few. In these works, multiple queries are merged in a very explicit way. For instance, in [10], the



Figure 11: The augmented **STORE*** dimension which includes the aggregated symbols.

table plan of each query is analyzed, and common accesses to the same table are merged into shared scans.

By merging query ranges into a common expression, we enjoy much of the same benefits; only they are a side-effect of the MDL-principle.

Retrieval of Aggregated Values: So far we assume that the only values one can retrieve are the lowest level values in an OLAP cube. However, quite often, the query fetches values that are aggregates along possibly multiple dimensions. For instance for the OLAP cube with the view *OLAPView*, one may fetch the total *Sales* and *SalesCount* of *Jacket* in the city of *NewYork*, so the range of the query contains the point (*Jacket*,*NewYork*) which is not part of the universe $\text{PRODUCT}_1 \times \text{STORE}_1$. We therefore cannot directly apply the OLAP query optimization techniques when aggregated values are involved in the query.

The solution is to enlarge the universe of the multidimensional structure to encompass the aggregations. Consider the *STORE* dimension shown in Figure 3 as a tree.

Suppose that the OLAP database rolls up the measures from the street level to the city level along the *STORE* dimension.

We first add new aggregated symbols: $\{\text{NewYork}^*, \text{SanFrancisco}^*, \text{Paris}^*\}$ to STORE_1 to form STORE_1^* and then redefine the structure as shown in Figure 11. Call this new hierarchical dimension **STORE***.

Now the new underlying structure is $\text{PRODUCT}_1 \times \text{STORE}_1^*$. One may repeat the same process for the *PRODUCT* dimension as well if the measures are rolled up along the *PRODUCT* dimension.

Performance Considerations:

It is clear that our proposed optimization technique requires additional and possibly intensive access to the dimensional structures. Often these structures are mapped to dimension tables, which, along with the fact table, are stored in a relational database. This may cause traversal in the dimension hierarchies to be costly.

We foresee a storage scheme for OLAP databases in which native data structures are used to index dimension tables. For example, tree-based indexing is used in [7]. The dimensions are usually of manageable size and slow varying, making it advantageous to index and distribute them among client machines to improve accessibility and availability. The fact table, however, is fast changing and can be potentially enormous in size, spanning multiple remote storage servers. With this in mind, it is reasonable to expend effort on optimizing OLAP queries by exploiting the dimensional structure.

6.2 Retrieval of text by keyword search in a XML document

Semistructured documents are often organized hierarchi-

cally, as in XML. Consider an XML document as a tree, where some nodes of the tree have lengthy text strings associated with them. For example, a “Chapter” node is associated with all the text in that chapter, a “Section” node with all the text in that Section, etc. The document is stored in such a form that given an identifier for a node, all the text associated with that node can be retrieved; for example, it might be stored by mapping the tree to a relational database. Suppose we wish to retrieve all the components of the document that contain a certain keyword. It is natural to assume that there is a full-text index that given a keyword can quickly locate the nodes whose text contains the keyword. We first retrieve from this index the—possibly large—set of nodes containing the keyword of interest. We then compute a \mathcal{L} -compact expression describing this set of nodes, using Theorem 4, and pose to the storage system the query that requests these nodes. The same advantages that we cited for the OLAP case, namely shorter queries that can be executed faster, are likely to occur here.

7. CONCLUSION AND FUTURE RESEARCH

We formalized the MDL problem for representations of subsets of structured sets. We studied the MDL problem for simple hierarchies and multidimensional hierarchies, and showed that the former can be solved in polynomial time while the latter is NP-complete for several different description languages.

We demonstrated that the MDL principle can be applied in query optimization for OLAP databases and XML documents. The resulting query is not only shorter, but can be executed more efficiently.

There are a number of directions for future work. First, experimental confirmation of the benefits to query optimization will be required.

Second, the NP-hardness of the multidimensional MDL problem calls for a detailed studied of heuristics for it. We outlined a simple approach using the existing polygon-covering heuristics, but it does not exploit the special features of the problem: that the dimensions of a multidimensional structure are categorical, not geometric, and that our objective is to minimize description length, not the number of hyperrectangles used. Finally, approximation bounds for the multidimensional MDL problem need to be studied.

Acknowledgement

This work was supported by the Natural Sciences and Engineering Research Council of Canada.

8. REFERENCES

- [1] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proc. SIGMOD*, pages 94–105. ACM Press, 1998.

- [2] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman, New York, 1979.
- [3] J. Gudmundsson and C. Levcopoulos. Close approximations of minimum rectangular covering. In *FST & TCS'96*, volume 1180 of LNCS, pages 135–146, 1996.
- [4] P. Kalnis and D. Papadias. Optimization algorithms for simultaneous multidimensional queries in OLAP environments. *Lecture Notes in Computer Science*, 2114:264–??, 2001.
- [5] R. Kimball. *The Data Warehouse Toolkit*. Wiley, 1996.
- [6] V. S. Anil Kumar and H. Ramesh. Covering rectilinear polygons with axis-parallel rectangles. In *Proceedings of the ACM Symposium on Theory of Computing*, pages 445–454, 1999.
- [7] L.V.S. Lakshmanan, R. T. Ng, C. X. Wang, X. Zhou, and T. J. Johnson. The generalized MDL approach for summarization. In *Proc. 28th VLDB Conference*, pages 445–454, 1999.
- [8] C. Levcopoulos and J. Gudmundsson. Approximation algorithms for covering polygons with squares and similar problems. In *RANDOM: International Workshop on Randomization and Approximation Techniques in Computer Science*. LNCS, 1997.
- [9] C. N. De Meneses and C. C. De Souza. Exact solutions of rectangular partitions via integer programming. *International Journal of Computational Geometry and Applications*, 10(5):477–522, 2000.
- [10] Y. Zhao, P.M. Deshpande, J.F. Naughton, and A. Shukla. Simultaneous optimization and evaluation of multiple dimensional queries. In *Proc. SIGMOD*, pages 271–282, 1998.