Cost-Based Labeling of Groups of Mass Spectra

ABSTRACT

We make two main contributions in this paper. First, we motivate and introduce a novel class of data mining problems that arise in labeling a group of mass spectra, specifically for analysis of atmospheric aerosols, but with natural applications to market-basket datasets. This builds upon other recent work in which we introduced the problem of labeling a single spectrum, and is motivated by the advent of a new generation of Aerosol Time-of-Flight Mass Spectrometers, which are capable of generating mass spectra for hundreds of aerosol particles per minute. We also describe two algorithms for group labeling, which differ considerably in how they utilize an LP solver, and also differ considerably from algorithms for labeling a single spectrum.

Our second main contribution is to show how to automatically select between these algorithms in a cost-based manner, analogous to how a relational query optimizer selects from a space of query plans. While the details are specific to the labeling problem, we believe that this is a promising first step towards a general framework for cost-based data mining, and opens up an important direction for future research.

1. INTRODUCTION

The size and composition of *aerosol particles*, which are often complex mixtures of organic and inorganic solids and liquid suspended in the air, is directly related to their origin, evolution and deposition and is intimately related to their environmental and health effects [19]. The *aerosol time-of-flight mass spectrometer (ATOFMS)* [20] samples aerosol particles directly from the ambient air or from an emission source and obtains size and chemical composition information on one particle at a time, in real-time. It holds the potential to fundamentally change policy and practice in environmental monitoring, but our ability to analyze the data is a critical bottleneck. Specifically, an ATOFMS produces a mass spectrum for each aerosol particle, and can sample about 250 particles per minute.

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.
Copyright 200X ACM X-XXXXX-XX-X/XX/XX ...$5.00.

A *mass spectrum* is a plot of *signal intensity* (often normalized to the largest peak in the spectrum) versus the *mass-to-charge (m/z) ratio* of the detected ions. Thus, the presence of a *peak* indicates the presence of one or more ions containing the corresponding m/z value. A basic task is to *label* a spectrum with the ions that are present in the particle, and we studied this problem in [10], in collaboration with a team of atmospheric chemists. In practice, however, we are often interested in the composition of particles sampled over some time window, rather than the composition of each individual particle.

In this paper, our first contribution addresses the central problem of labeling a group of mass spectra. To a first approximation, we treat the group as an unordered collection. The fact that these spectra are obtained from a continuously sampled stream of particles essentially allows us to exploit some domain knowledge about the percentage of similar spectra within a group. The labeling of mass spectra is a first step in a more comprehensive analysis of ATOFMS streams, and allows us to model each aerosol particle as a collection of ions, along with a quantity for each ion. Viewed thus, an aerosol particle is a generalization of the well-known market basket abstraction of a collection of items purchased at one time by a customer. While our primary focus is on mass spectra, we briefly discuss this connection to market basket data, which makes our results relevant for a wider class of applications.

Additional steps in a typical analysis involve looking for trends and correlations with other spatiotemporal streams, taking into account data about ambient conditions and emission sources. Thus, labeling is just one step in a typical multi-step analysis. Our ultimate objective is to develop an algebraic framework for expressing such multi-step data mining analyses, and a cost-based optimization framework for finding good evaluation plans.

Our second contribution in this paper is to show how database concepts like set-orientation and cost-based query optimization can be applied to data mining tasks (such as labeling mass spectra using linear programming techniques). While the details are specific to the labeling problem, we show the benefits of a set-oriented approach to a complex mining task, in particular, the benefits of essentially "pushing" constraints over the desired set of labels down into the linear programming computations for identifying those labels. The cost analysis of the algorithms we propose for group labeling clearly shows the benefits of group labeling versus labeling each spectrum in the group individually. Most importantly, it provides the basis for a cost-based

approach to selecting the most efficient algorithm. To our knowledge, this is the first paper to describe a cost-based framework for selecting between alternative data mining algorithms (including algorithms for machine learning problems, statistical analyses, various kinds of frequent itemset and sequential pattern identification, etc.).

Research in data mining has largely concentrated on algorithms for a single task, and comparisons of performance have been empirical in nature. Insights from database systems design can guide the development of a framework for multi-step analyses that incorporates data mining tasks such as clustering, decision-tree construction, or labeling. In turn, this opens the door to a cost-based optimization framework, and to a compositional approach to mining. We believe that our results are a modest first step towards this goal.

1.1 Outline

The rest of the paper is structured as follows. We review single spectrum labeling in Section 2 and then introduce group labeling in Section 3. We propose two new algorithms for group labeling in Section 4. In Section 5, we present a cost analysis of these algorithms. Using this analysis as a foundation, we propose a cost-based method for selecting the most efficient group labeling algorithm, taking into account the characteristics of the data and the group labeling parameters, in Section 6. In Section 7, we study the algorithm selection method experimentally, and show its effectiveness. In Section 8, we discuss the connections between spectrum labeling and market basket analysis. We survey related work in Section 9.

2. SPECTRUM LABELING

In this section, we review the problem of labeling a single mass spectrum, introduced in [10], to keep this paper self-contained. The formalization of the group labeling problem, presented in the next section, builds upon the single spectrum case.

2.1 Preliminaries

A **mass spectrum** can be represented as a normalized vector $\vec{b}$, $\sum_i b_i = 1$. $b_i \in \mathbb{R}$ is the relative signal intensity at m/z value i.

The **signature** of an ion is a vector $\vec{s}$, $s_i \in \mathbb{R}$ and $\sum_i s_i = 1$, representing the distribution of its isotopes, i.e., s_i is the relative abundance of isotopes with m/z value i.

A **signature database** is a set of signatures $S = \{\vec{s_1}, \vec{s_2}, ..., \vec{s_n}\}$ where $\vec{s_j}$ is the signature of chemical element j. All the spectra and signatures have the same 'range' and 'granularity' over m/z axis; i.e., they have the same dimension and the i^{th} element of a spectrum or signature always corresponds to the same m/z value i.

The task of **spectrum labeling** is to find the chemical ions identified by the peaks in the spectrum and, ideally, their quantities in the particle. If we arrange the n signatures in the signature database in some order, the signature database can be represented as a matrix $A = [\vec{s_1}, \vec{s_2}, ..., \vec{s_n}]$, where the k^{th} column in matrix A represents signature k. The labeling task consists of finding an n-dimensional vector $\vec{x}$ such that $\vec{x}[j]$ is the relative abundance of chemical element j. This is equivalent to solving the linear equation

$$A\vec{x} = \vec{b}, \vec{x} \geq 0. \quad (1)$$

2.2 An Optimization-Based Reformulation

2.2.1 Error Bound

In real applications, the observed spectrum usually contains noise and calibration discrepancies, and cannot be described as an exact linear combination of ion signatures. Labeling therefore involves finding a linear combination of ion signatures that approximately matches the input spectrum. Therefore, we introduce an error bound E with respect to a certain distance function D. The linear equation model (1) then becomes an optimization task:

$$Seek \quad \vec{a}, \quad s.t. \quad D(A\vec{a}, \vec{b}) < E, \quad \vec{a} \geq 0 \quad (2)$$

Given a signature database A that contains n signatures, and an input spectrum $\vec{b}$, the search space for the optimization task defined in (2) is an n-dimensional continuous space. The **solution space** for input $\vec{b}$ is a subspace within this search space.

Definition 1. Given a signature database A, an input spectrum $\vec{b}$, and an error bound E with respect to distance function D, the **solution space** of spectrum $\vec{b}$,

$$\mathfrak{L}_{\vec{b}} = \{\vec{a} \mid D(A\vec{a}, \vec{b}) < E \text{ and } \vec{a} \geq 0\}$$

It is worth noticing that the choice of the distance function D in (2) could dramatically change the complexity of the problem [10]. Using Manhattan distance, namely $D(\vec{v_1}, \vec{v_2}) = \sum_i |\vec{v_1}[i] - \vec{v_2}[i]|$, the optimization task of (2) can be interpreted as a linear programming task [10] whose time complexity is polynomial in the total number of signatures in the signature database. Other distance functions can be useful in certain situations, for example, to spread errors over fewer dimensions. However, considering other distance functions is outside the scope of this paper, and we will henceforth assume that Manhattan distance is used.

2.2.2 Optimization Model

In [10], we have shown that the optimization task defined in (2) will have an infinite number of solutions for most input spectra. Fortunately, in practice, we only care about those solutions that are significantly different. A natural approach to deal with the infinity in a continuous space is to discretize it into grids, so that the number of possible solutions is finite.

Formally, a discretization is specified by a **threshold vector** $\vec{t} = [t_1, t_2, ...t_{d+1}]$ divides each dimension of the search space into d ranges: $[t_1, t_2), [t_2, t_3)...[t_d, t_d + 1)$, where t_i and t_{i+1} are the lower bound and upper bound of range i. A **cell** is the finest granularity of the discretization, which characterizes the degree of detail users care about. Given a discretization that divides each dimension into d ranges, the whole search space is discretized into d^n *cells*, where n is the number of dimensions. (Recall that n is the number of signatures in the signature database.)

A **label** of spectrum $\vec{b}$ is simply a cell that intersects $\vec{b}$'s solution space. It can be represented as a vector of integers $\vec{x}$, s.t. $\vec{x}[i]$ indicates the range it falls into on dimension i.[1] The set of all cells intersecting $\vec{b}$'s solution space forms the **label set** of spectrum $\vec{b}$. We use the term **feasible**

[1]This is defined rigorously using the notion of an **index vector** in [10].

subspace to describe any subspace of the search space that intersects the solution space.

Figure 1 illustrates the concepts discussed in this section. Suppose there are two signatures in the signature database. The threshold vector $\vec{t} = [0, 0.3, 0.6, 1]$ divides each dimension into three ranges indexed by 0, 1, and 2. The search space is a two-dimensional space $ABCD$. $S_1S_2S_3S_4$ is the solution space of an example input spectrum, which intersects the *cells* $LFGM$ and $MGHA$. So, *cells* $LFGM$ and $MGHA$, which can be represented as vectors $[0, 1]$ and $[0, 2]$, are labels of the input spectrum. Subspace $ALFH$ intersects the solution space, so it is a *feasible subspace*. $MBEG$ is also a *feasible subspace*.

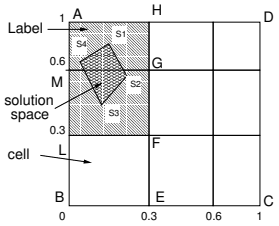

Figure 1: Illustration of Concepts

Given an error bound E with respect to a distance function D and a discretization, we now redefine the task of **spectrum labeling** as follows: *Find all cells that intersect the solution space of the input spectrum.*

Table 1 summarizes the notations used in this paper and provides an operational optimization model for the labeling task that we just described.

Notation:	$\vec{x}$	A n dimensional vector of integers , $1 \leq \vec{x}[i] \leq d$.
	$\vec{b}$	Input mass spectrum
	$\vec{t}$	Threshold vector for discretization
	d	Number of ranges per dimension under discretization
	L	Label set of input spectrum
	A	Signature database
	D	Distance function
	E	Error bound
L=$\emptyset$		
For every possible $\vec{x}$, $1 \leq \vec{x}[i] \leq d$		
Seek $\vec{a}$ s.t.		
$D(A\vec{a}, \vec{b}) \leq E$ (3)		
$\vec{t}[j] \leq \vec{a}[i] < \vec{t}[j+1]$, $j = \vec{x}[i]$		
If (3) succeeds, $L = L \cup \vec{x}$		
Return L		

Table 1: Operational Definition of Spectrum Labeling

3. GROUP LABELING

In this section, we introduce the problem of labeling a group of spectra. In environmental monitoring, the spectra are collected through continuous sampling, and a group that is collected at a single location over a short time-span is likely to contain many similar spectra (because the environment does not change instantaneously). Thus, the goal is to find these common, or typical, spectra.[2] Indeed, this is the goal even when the group does not reflect particles from the same location and time; e.g., when analyzing a collection of spectra obtained at multiple locations and times but with some commonalities in ambient conditions.

Given a group of spectra $\{\vec{b_i}\}$, we can conceptually[3] compute a set of label sets $\{L_i\}$, where $L_i = \{\vec{x_{ij}}\}$ is the label set of $\vec{b_i}$. We define the **support** of a label $\vec{x}$ with respect to the group of spectra $\{\vec{b_i}\}$ as the percentage of $\vec{b_i}$s whose corresponding label set contains $\vec{x}$.

Definition 2. Given a group of spectra $\{\vec{b}\}$, the **support** of a label $\vec{x} = \frac{|\{L_i | \vec{x} \in L_i\}|}{|\{L_i\}|}$, where $L_i = \{\vec{x_{ij}}\}$ is the label set of spectrum $\vec{b_i}$.

Intuitively, the support characterizes the likelihood of a label given a group of similar spectra. Extending the concept of 'label' and 'label set' discussed in Section 2.2, we define **group label** and **group label set** as follows:

Definition 3. Given a group of spectra $B = \{\vec{b_i}\}$ and a threshold Min_Sup, $\vec{x}$ is a **group label** if the *support* of $\vec{x}$ w.r.t. B is greater than Min_Sup. The **group label set** for the group B is $GL = \{\vec{x} | \vec{x}$ *is a group label of* $B\}$.

As an example, consider a group of spectra $\{\vec{b_1}, \vec{b_2}, \vec{b_3}\}$. Let the label set for $\vec{b_1}$ be $\{\vec{x_1}, \vec{x_2}\}$, the label set for $\vec{b_2}$ be $\{\vec{x_2}\}$, and the label set for $\vec{b_3}$ be $\{\vec{x_3}\}$. Then, $support(\vec{x_1}) = support(\vec{x_3}) = 33\%$, $support(\vec{x_2}) = 66\%$. Suppose the Min_Sup threshold is set to be 50%, then $\vec{x_2}$ is the only *group label*. The *group label set* is therefore $\{\vec{x_2}\}$.

Spectral labeling is important in many domains other than environmental monitoring because mass spectra are a widely used tool for chemical and biological analysis. Surprisingly, the concepts also show promise for analyzing market-basket data; we discuss this briefly in Section 8.

4. SEARCH FOR GROUP LABELS

In this section, we first review a depth-first search algorithm introduced in [10] for single spectrum labeling, based on which we propose two new algorithms for group labeling. When we go from labeling a single spectrum to a large group of spectra, the problem is fundamentally altered by the notion of support. The new Depth First Search with Voting (DFSVoting) and Candidate Generation and Test(GenTest) algorithms for group labeling differ significantly in how they handle support.

4.1 Basic Depth First Search Algorithm

If a subspace is not feasible, then we do not need to consider any cell in that subspace. The basic depth-first single spectrum labeling algorithm utilizes this property to prune the search space. Table 2 shows the operational procedure which invokes an LP call to test whether a given subspace is feasible.

[2]A related task is to find common ions across the group of spectra. Further, we often have domain knowledge that can be expressed in terms of constraints over the composition of the particles in the group. These extensions are important directions for future research, but outside the scope of this paper.

[3]Computing all label sets is inefficient, and the group labeling algorithms that we propose avoid this.

Given :	Input spectrum $\vec{b}$ Threshold vector $\vec{t}$ Error bound E
is_feasible(subspace S) Seek $\vec{a}, s.t.$ $D(A\vec{a}, \vec{b}) \leq E$ (*) $\vec{t}[l_i] \leq \vec{a}[i] < \vec{t}[h_i]$ $\vec{t}[l_i]$ and $\vec{t}[h_i]$ are the boundary of S in dimension i if (*) succeeds, return TRUE, otherwise return FALSE	

Table 2: Testing the Feasibility of a Subspace

The basic depth-first single spectrum labeling algorithm is shown in Table 3, and uses a divide-and-conquer approach. Its exploration of the search space can be mapped to a search tree. Each node in the search tree is associated with a unique subspace.

At each node, the algorithm first invokes a linear programming (LP) call to check if the subspace is feasible. If the subspace is not feasible, the subtree is pruned and not explored. Otherwise, we know there are one or more labels in the subspace, and we must search inside that subspace. To do this, we select a dimension j that has not been subdivided to the finest possible granurality, and use it to split the subspace into smaller spaces, each of which has the finest possible granurality in dimension j. Each smaller space created thus corresponds to a new search node is, and is explored recursively.

The above procedure is repeated until either (1) the current subspace is not feasible, or (2) the current subspace is a cell. In the former case, we discard the current search node and backtrack. In the latter case, the label corresponding to this cell is output by the algorithm.

Given :	Input spectrum $\vec{b}$ Threshold vector $\vec{t} = [t_1, t_2, \ldots, t_{d+1}]$ Error bound E
Output:	Label set for $\vec{b}$

Depth_First_Search(subspace S)
if (is_feasible(S))
RETURN
else
if (S is a cell)
output the corresponding label of S
else
pick_dimension(j)
split S into a set of subspacesS_i
s.t. Each S_i is not divisible on dimension j
for each result subspace S_i
Depth_First_Search(S_i)
 	
Main: *Depth_First_Search*(the whole search space W)	

Table 3: Algorithm for Single Spectrum Labeling

In Table 2, the method *pick_dimension*(j) chooses the dimension to split. We use a simple scheme in which $(k+1)^{th}$ dimension is chosen as the split dimension at level k of the recursion, assuming the search starts from level 0.[4]

[4]Different strategies for choosing the dimension to split are studied in [10].

4.2 Depth-First Search Voting Algorithm

In group labeling, a subspace is 'feasible' (i.e., worth further exploration) only when it intersects the solution spaces of at least a certain minimum number of spectra. Following this intuition, we derive the DFSVoting group labeling algorithm (shown in Table 4) from the depth-first single spectrum labeling algorithm by changing the definition of 'feasible'.

At each search node, we take a vote among the spectra in the group. A spectrum votes *yes* at a node if the subspace corresponding to the node is feasible for the spectrum; otherwise it votes *no*. When the number of *yes* votes exceeds the minimum number required by the support theshold, the algorithm goes on to search the children of the current node in depth-first order. Otherwise, the subspace at the current node is pruned, and the algorithm backtracks to the parent node.

Consider an example of group labeling, with two signatures in the database and with two spectra in the group, and the threshold vector for discretization set to be $\vec{t} = [0, 0.3, 0.6, 1]$. Suppose that the label sets for the two spectra are $\{\vec{x_1}, \vec{x_2}, \vec{x_3}\}$ and $\{\vec{x_1}, \vec{x_2}, \vec{x_4}\}$ respectively, in which $\vec{x_1} = [0, 1]$, $\vec{x_2} = [0, 2]$, $\vec{x_3} = [0, 0]$ and $\vec{x_4} = [1, 2]$.

Figure 2 illustrates the execution of the DFSVoting algorithm. The shadow area in each search node represents the subspace investigated. Beside each search node, we show the set of spectra that vote *yes* for the subspace, and the order in which nodes are visited. The edge connecting two search nodes is tagged by the additional constraint introduced when going from the parent to its child.

Input :	Set of Spectra B, $\|B\| = w$ Threshold vector $\vec{t} = [t_1, t_2, \ldots, t_{d+1}]$ Error bound E Support threshold Min_Sup
Output:	Group label set for B

DFSVoting(Subspace S, Set of Spectra C)
$C' = \{\vec{b} \vec{b} \in C, S$ is feasible w.r.t. $\vec{b}\}$
if $\|C'\| \leq Min\_sup * w$
RETURN
else
if (S is a cell)
output the corresponding label of S
else
pick_dimension(j)
split S into a set of subspaces S_i
s.t each S_i is not divisable on dimension j
for each result subspace S_i
DFSVoting(S_i, C')
 	
Main : *DFSVoting*(The whole search space W, B)	

Table 4: Algorithm DFSVoting for Group Labeling

The following theorem establishes the correctness of DFSVoting. The proof is omitted for lack of space.

THEOREM 1. *Given a group of spectra and a specified minimum support, the DFSVoting algorithm finds the complete group label set without duplication.*

4.3 Candidate Generation and Test Algorithm

The Candidate Generation and Test (GenTest) algorithm

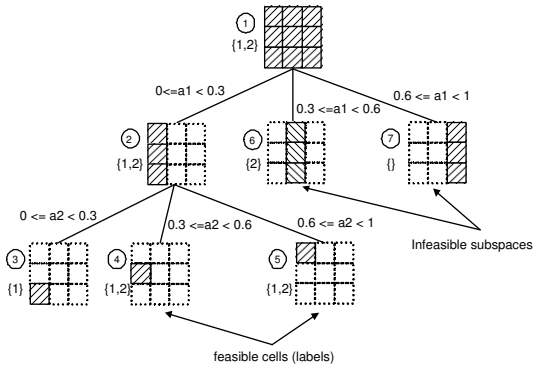

Figure 2: An Example of the DFSVoting Algorithm

uses the depth-first algorithm for single spectrum labeling as a building block. It is based on the following observation:

LEMMA 1. *Suppose that we are given a support threshold Min_Sup, a set of spectra $B, |B| = w$, and a subset S, $S \subset B$, $|S| = \lfloor (1 - Min\_Sup) * w + 1 \rfloor$. Then, l is a group label $\Rightarrow \exists \vec{b} \in S$, s.t. l is a label of $\vec{b}$.*

The above lemma essentially uses a pigeon-hole argument to establish that labels with a given level of support can only be missing in a certain (hopefully small, for high support) number of spectra. In particular, such a label must appear in the labels for some spectrum in set S if we pick $|S| = \lfloor (1 - Min\_Sup) * w + 1 \rfloor$. Thus, the union of the label sets of spectra in such a set S contains all group labels for B.

The GenTest algorithm shown in Table 5 consists of two phases: (1) Select a group S with $\lfloor (1 - Min\_Sup) \times w + 1 \rfloor$ spectra from B and calculate the label set for each of them. This generates a set of candidates group labels. (2) For each candidate group label, test whether it is a label for each spectrum in $B - S$. If a candidate label appears in the label set of at least $\lceil w \times Min\_Sup \rceil$ spectra, it is output as a group label for B.

The following theorem establishes the correctness of GenTest.

THEOREM 2. *Given a group of spectra and a specified minimum support, the GenTest algorithm finds the complete group label set without duplication.*

We observe that both algorithms are highly parallelizable. DFSVoting is also non-blocking, in contrast to GenTest, in which the testing phase is blocked until the candidate generation phase is complete. A more detailed analysis that compares the cost of the two algorithms is presented in the next section.

5. COST ANALYSIS

The goal of our analysis is to estimate the effect of various inputs on the overall cost of each algorithm, and more importantly, to determine the relationship between algorithm cost and the characteristics of the data. Ultimately, we want to be able to select the less expensive algorithm for any instance of the problem by using these cost estimates.

In what follows, we will use the notation in Table 6.

Input :	Set of Spectra B, $\|B\| = w$
	Threshold vector $\vec{t} = [t_1, t_2, \ldots, t_{d+1}]$
	Error bound E
	Support threshold Min_Sup
Output:	Group label set for B

GenTest

```
L = ∅
B_0 = {⌊(1 − Min_Sup) * w + 1⌋ spectra
       randomly choosen from B}
for each b⃗ in B_0
    find F_i, the label set of b⃗
    for each label l ∈ F_i
        l.count++
    L = L ∪ F_i
for each spectrum b⃗ ∈ B − B_0
    for each l ∈ L
        if l is a label for the b⃗
            l.count++;
for each label l ∈ L
    if (l.count > Min_Sup * w)
        output l as a solution
```

Table 5: Algorithm GenTest for Group Labeling

Notation	Meaning
n	Number of element signatures in the database
d	Number of ranges per dimension under discretization
m	Number of labels for a particular spectrum
w	Size of the group of spectra
s	Conceptual number of identical spectra within the group
Min_Sup	Minimum support threshold for group labeling
C_{Single}	Cost of labeling a single spectrum
C_{Voting}	Cost of DFSVoting algorithm
$C_{GenTest}$	Cost of GenTest algorithm

Table 6: Notation for Cost Analysis

5.1 Cost Metric

The proposed algorithms call an LP solver to determine whether a subspace is feasible or not. The exact cost of an LP call depends on the initial point and the constraints. When more sophisticated optimization is used, the cost of a particular LP call may also depend on the previous linear programming tasks performed [18]. Fortunately, the cost of one LP call is polynomial in the number of signatures in the database [18] and both DFSVoting and GenTest tend to have similar gains when given additional constraints and similar input spectra. Therefore, the number of LP calls incurred is a good cost metric, at least for comparing the two algorithms. In addition, this abstraction makes our analysis applicable to other depth-first search algorithms that invoke expensive subcomputations at each node.

5.2 Cost of Labeling One Spectrum

The depth-first single spectrum labeling algorithm takes a spectrum as input and outputs its label set. The search space corresponds to a complete tree, as shown in Figure 3.

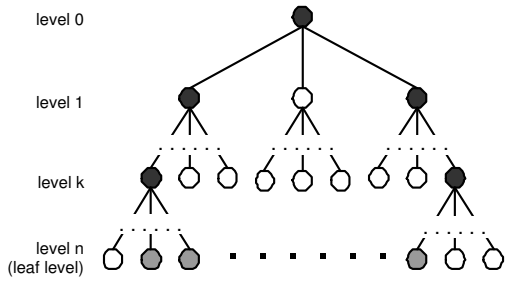

Figure 3: A Complete Search Tree

Each leaf node corresponds to a cell in the space. The tree is traversed in a top-down fashion. At each non-leaf node visited, we invoke an LP call to see if it has in its subtree a leaf node corresponding to a *label*. If there is such a leaf node, all the children of the subtree are visited, otherwise, the algorithm will prune that subtree. We can think of the algorithm as a **node coloring game**.

> *Given a complete tree of $n+1$ levels, a painter **randomly drops** m black balls on the leaf nodes and colors the non-leaf nodes as follows: If a non-leaf node has a black ball in its subtree, paint it black; otherwise, leave it white.*

The painter corresponds to the input spectrum. The number of black balls is the number of labels for that spectrum. The complete tree with $n+1$ levels corresponds to the entire search space of the depth-first algorithm, and each leaf node with a black ball is a label. A non-leaf node is black if its corresponding subspace is feasible.

LEMMA 2. *Given a spectrum $\vec{b}$ and a discretization criterion that divides each dimension of the search space into d ranges, if the corresponding* node coloring game *ends with n_b black non-leaf nodes, the number of LP calls invoked by the depth-first algorithm in Table 3 to label spectrum $\vec{b}$ is:*

$$C_{single} = n_b * d \ + 1 \tag{4}$$

PROOF. In the node coloring game, a non-leaf node is black if it corresponds to a feasible subspace. Black nodes are those that invoke one LP call for each of their children. Since each black node invokes an LP call for each of its children and one LP call is performed at the root node, the total number of LP calls invoked is $n_b * d \ + 1$. □

In transforming the single spectrum labeling problem to a node coloring game, we deliberately made a **random drop assumption**: *A label of a spectrum is randomly and uniformly assigned to a cell in the search space.*

This assumption actually allows duplicates among the m labels for a spectrum, which is not the case for our labeling algorithm. However, the number of labels m is much smaller than the total number of cells d^n. Therefore, the difference due to duplicates is negligible.

In reality, the uniform distribution assumption is also violated. The labels are not spread out in the space without any constraints. They all intersect the solution space of the input spectrum, which is a convex hull [10]. In other words, they are close to each other in the space. Nonethless, these simplifications allow us to derive cost formulae that track actual performance very well, as we show empirically in Section 7.

Given the 'node coloring game' model, to estimate cost, we have to estimate the number of black non-leaf nodes. In order to estimate the total number of black nodes after playing the game, we first estimate the probability that a particular non-leaf node is painted black.

LEMMA 3. *In the node coloring game, if the painter has m balls, the probability that a non-leaf node at level k is colored black is:*

$$P(k) = 1 - \left(1 - \frac{1}{d^k}\right)^m \tag{5}$$

PROOF. At a particular level k, there are d^k nodes. So, for a particular non-leaf node N at level k the probability that a particular black ball is in the subtree of N is $\frac{1}{d^k}$. $1 - \frac{1}{d^k}$ then gives the probability that a particular black ball is not in the subtree of N. Since each ball is dropped independently, the probability that all m black balls are not in N's subtree is $\left(1 - \frac{1}{d^k}\right)^m$. Hence, $1 - \left(1 - \frac{1}{d^k}\right)^m$ gives us the probability that node N at level k has a black ball in its subtree. In other words, the probability that a painter colors a node black at level k is:

$$P(k) = 1 - \left(1 - \frac{1}{d^k}\right)^m$$

□

Given the function P in Lemma 3, we can estimate the number of black nodes at level k, and in turn, the overall number of LP calls invoked by the depth-first algorithm for labeling a single spectrum.

THEOREM 3. *Given a signature database with n signatures and a threshold vector $\vec{t}$ that divides each dimension of the search space into d ranges, under the random drop assumption, the expected number of LP calls invoked by the depth-first algorithm shown in Table 3 to label a single spectrum with m labels is:*

$$C_{Single} = d * \left(\sum_{k=0}^{n-1} d^k * \left(1 - \left(1 - \frac{1}{d^k}\right)^m\right)\right) + 1 \tag{6}$$

PROOF. The spectrum labeling process is equivalent to the node coloring game. The complete search tree as shown in Figure 3 has n levels (counting from 0). Each non-leaf node has d children. For each level k, there are d^k equivalent nodes. According to Lemma 3, the probability of a non-leaf at level k being black is P(k), so the average number of black nodes at level k is $d^k * P(k)$. Adding up the number of black nodes at each non-leaf level gives us the number of black nodes in the tree: $\sum_{k=0}^{n-1} d^k * P(k)$. Combining the result of Lemma 2, we have the total number of LP calls invoked by the basic depth first search algorithm as: $C_{Single} = d * \left(\sum_{k=1}^{n-1} d^k * P(k)\right) + 1$. Replacing P(k) with the formula given in Lemma 3 leads to the formula (6) stated in this theorem. □

When k is large, $P(k)$ is reduced to $\frac{m}{d^k}$ and formula (6) is approximately equivalent to:

$$C_{Single} \approx d \ * \ (n-1) \ * \ m \tag{7}$$

This suggests that the number of LP calls invoked by the algorithm is linear in the number of labels for the input spectrum.

5.3 Cost of Group Labeling

When we go from single spectrum labeling to labeling a group of spectra, the analysis is complicated further by the fact that data distribution has a significant impact on performance. In this subsection, we first propose a simple model to characterize data distribution, and then analyze the cost of the two group labeling algorithms. Our analysis of the relationship between data distribution and algorithm cost, leads to the discussion of cost-based algorithm selection in Section 6.

5.3.1 A Model of Data Distribution

As discussed in Section 3, the majority of spectra in groups that we want to label tend to be very similar to each other. A simplified way to model this is that *most spectra in a group are identical, while the rest are random noise or 'impurities' with great variance*. Following this intuition, we model a group of w spectra as s identical spectra mixed with $w - s$ random 'noise' spectra which are greatly different from each other.

While this is an overly simplified model of the data, note that the number of identical spectra s is just a conceptual parameter which describes the 'diversity'(or 'variance') of the data. Of course, more complicated statistical tools, such as Chi-Square testing [4] and other deviation detection and characterization methods [2] can be adopted for characterizing the data. The simple model we propose, however, suffices for a cost analysis aimed at estimating the relative performance of DFSVoting and GenTest.

5.3.2 DFSVoting

The DFSVoting algorithm proposed in Section 4 is a direct extension of the depth-first algorithm for single spectrum labeling. All the analysis for the single spectrum case still holds, with the difference that we now have a group of painters voting for the color of a non-leaf node.

According to the notation in Table 6, we have a group of w spectra, within which s spectra are the same. The group labeling algorithm will look for all the labels that are common to at least $t = \lceil w * Min\_Sup \rceil$ spectra. We again assume each spectrum has m labels.[5] The *node coloring game* for single spectrum labeling then becomes the **group node coloring game** described below.

> *There are w painters in the game, and each has m black balls. They randomly drop the balls onto the leaf nodes. For a particular node N, a painter votes* yes *if at least one of his black balls is in N's subtree. A node is painted black if at least $t = \lceil w * Min\_Sup \rceil$ painters vote* yes.

As described in Section 4, if a node has v votes ($v \geq t$), it is painted black and v LP calls are issued for each of its children; otherwise, the node is 'white', and is pruned. In addition, the root node requires w LP calls. The cost of DFSVoting is therefore:

$$C_{Voting} = w + d * \#Votes\ got\ by\ all\ black\ nodes \quad (8)$$

LEMMA 4. *In the group node coloring game, if there w painters* ***independently*** *vote for the color of the nodes, the probability that a particular node at level k receives v votes is:*

$$PVote(v, w) = \mathrm{C}_w^v \ * \ P(k)^v * (1 - P(k))^{w-v} \quad (9)$$

PROOF. As shown in Theorem 3, for a non-leaf node at level k, the probability that a painter drops at least one black ball in N's subtree is $P(k)$. Thus, with a probability $P(k)$, node N will get a vote from a particular painter. Since all the painters make independent decisions, given a group of w painters, the probability that node N receives v votes is:

$$PVote(v, w) = \mathrm{C}_w^v \ * \ P(k)^v * (1 - P(k))^{w-v}$$

□

Lemma 4 studies the situation when painters make decisions *independently*. We now extend it to the case when some of them always make the same decision, and in turn estimate the number of LP calls invoked by a particular node.

LEMMA 5. *Following the notation in Table 6, let $t = \lceil w * Min\_Sup \rceil$. Given a group of w spectra of which s are identical, under the random drop assumption the expected number of LP calls invoked at a particular non-leaf node at level k in the search tree is:*

$$\begin{aligned} &NodeCost(k) = P(k) * \\ &\left(\sum_{v=max(t,s)}^{w} \mathrm{C}_{w-s}^{v-s} P(k)^{v-s} (1 - P(k))^{w-v} * d * v\right) + \\ &(1 - P(k)) * \left(\sum_{v=t}^{w-s} \mathrm{C}_{w-s}^{v} P(k)^{v} (1 - P(k))^{w-s-v} * d * v\right) \end{aligned} \quad (10)$$

PROOF. When there are s identical spectra in the group of w spectra, the DFSVoting algorithm described in Section 4 will act as if s painters out of w are 'identical' (making exactly the same decision all the time), which means a node will either get all the votes of those s painters or lose all their votes. Apart from the s identical painters, the remaining painters still vote independently, as before. According to Lemma 4, the probability that node N receives v_1 votes from the remaining $w - s$ painters is $PVote(v_1, w - s)$. If s identical painters all vote for node N, then the probability of node N receiving v ($w \geq v \geq s$) votes is $PVote(v - s, w - s)$. If (and only if) a node receives v votes, $v \geq t$, we invoke v LP calls for each of its d children. Thus, the expected number of LP calls invoked at node N under the precondition that s identical painters all vote for N is:

$$E_{yes} = \sum_{v=max(t,s)}^{w} PVote(v - s, w - s) * d * v \quad (11)$$

Similarly, the expected number of LP calls invoked by node N under the precondition that s identical painters all vote *no* at node N is:

$$E_{no} = \sum_{v=t}^{w-s} PVote(v, w - s) * d * v \quad (12)$$

Since s identical painters act alike, they vote *yes* at node N with probability $P(k)$, and vote *no* with probability $1 - P(k)$. Combining this with formula (11) and formula (12), we arrive at the overall estimated number of LP calls invoked at node N: $P(k) * E_{yes} + (1 - P(k)) * E_{no}$, which is the same as formula 10 given in this theorem. □

[5] This is a strong assumption. If spectra differ a lot, the size of label sets may vary greatly. However, when the majority of spectra are similar, this is a reasonable simplification.

This leads us to the formula estimating the overall cost of the DFSVoting algorithm.

THEOREM 4. *Assume there are w spectra in the group, each of which has m solutions, and that s of them are identical. Under the random drop assumption, given a signature data base of n signatures, and a discretization that divides each dimension of the search space into d ranges, the expected number of LP calls in the DFSVoting algorithm is:*

$$C_{Voting} = w + \sum_{k=0}^{n-1} d^k * NodeCost(k) \tag{13}$$

.

PROOF. Given a signature database of n signatures, the search tree of DFSVoting has n levels. There are d^k equivalent nodes at level k. According to Lemma 5, a particular node at level k will invoke $NodeCost(k)$ LP calls. So the expected number of LP calls invoked at level k is $d^k * NodeCost(k)$. Adding the LP calls invoked at each non-leaf level plus the w LP calls at the root gives us an estimate of the total number of LP calls invoked by the algorithm: $C_{Voting} = w + \sum_{k=0}^{n-1} d^k * NodeCost(k)$. □

5.3.3 GenTest

THEOREM 5. *Suppose that randomly selected spectra from a group follow the same data distribution as the group. Following the notation in Table 6, if we are given a signature database of n signatures, and a discretization that divides each dimension of the search space into d ranges, under the random drop assumption the expected number of LP calls invoked by the GenTest algorithm is:*

$$\begin{aligned} C_{GenTest} \approx\ & d * (n-1) * m * (w-t+1) + \\ & \big((w-t+1) * (1-\tfrac{s}{w}) + 1\big) * m * (t-1), \end{aligned} \tag{14}$$

where $t = \lceil w * Min\_Sup \rceil$.

PROOF. The GenTest algorithm described in Section 4 has two phases, and we analyze the cost of each phase below.

Candidate Generation Phase: GenTest selects $w-t+1$ spectra from the group. For each of these spectra, it generates a label set. The cost of searching for the label set for a single spectrum is C_{Single}, from Theorem 3. Therefore, the number of LP calls invoked for generating the candidate labels is:

$$C_{Single} * (w-t+1) \tag{15}$$

Test Phase: The GenTest algorithm takes every generated candidate label and tests it on the remaining $t-1$ spectra. In the generation phase, $(w-t+1)$ spectra are randomly selected. According to the assumption that the randomly selected spectra follow the same data distribution as the original group, there will be $\frac{s}{w} * (w-t+1)$ identical spectra with the same m labels and $(1-\frac{s}{w}) * (w-t+1)$ spectra that have distinct labels. So the total number of candidates generated will be $\big((1-\frac{s}{w}) * (w-t+1) + 1\big) * m$. Since each 'test' invokes an LP call, the total number of LP calls invoked in this phase is the number of candidates times the number of remaining spectra:

$$(t-1) * m * \Big((1-\frac{s}{w}) * (w-t+1) + 1\Big) \tag{16}$$

Adding the cost of the generation and test phases gives us the total number of LP calls invoked by GenTest:

$$\begin{aligned} C_{GenTest} = & C_{Single} * (w-t+1) + \\ & (t-1) * m * \big((1-\tfrac{s}{w}) * (w-t+1) + 1\big) \end{aligned} \tag{17}$$

Substituting C_{Single} with equation (7), we have $C_{GenTest} \approx d*(n-1)*m*(w-t+1)+(t-1)*m*\big((1-\frac{s}{w}) * (w-t+1) + 1\big)$. □

6. ALGORITHM SELECTION

We can use the cost formulae for DFSVoting and GenTest to estimate the cost of both DFSVoting and GenTest. Evaluating these formulae at optimization time has two drawbacks: (1) The calculation of C_{Voting} involves very high precision floating point arithmetic, which is rather costly. (2) It is hard to tune the cost estimates in cases when there is significant discrepancy between the estimates and the observed real cost.

In this section, we propose an approach to algorithm selection that relies on precomputing *decision plots*, which essentially capture the performance tradeoffs between algorithms. The precomputation approach also highlights an important point: *Even if closed-form formulae cannot be derived to accurately predict algorithm costs, unlike the case for DFSVoting and GenTest, a promising approach is to start with a rough initial estimate for a decision plot and to apply machine learning or statistical modeling techniques to refine the estimate.*

6.1 Algorithm Profile

Using the cost analysis discussed in Section 5, we can plot the relation between Min_Sup and cost for a given group of spectra, assuming that the number of identical spectra in the group, s, is also known or can be estimated. Figure 4 shows a series of graphs derived from the calculation of formulae (13) and (17). The y-axis in those graphs is the estimated number of LP calls invoked by the algorithm. The x-axis represents the Min_Sup value specified by the user. These graphs characterize the performance characteristics of the two algorithms with respect to input data. We call such a graph an **algorithm profile**. We focus on the case when the group size is fixed, for simplicity; otherwise, this adds an extra dimension to the algorithm profile.

The group size in the *algorithm profile* shown in Figure 4 is set to be 1000. Each graph in the series corresponds to a particular s value shown in the upper right corner. As we can see in the *algorithm profile*, the lines for DFSVoting and GenTest intersect, which indicates that the choice of Min_Sup will change the algorithm of choice for a given group of spectra. From the series of graphs shown in Figure 4, we also notice that the intersection point varies with s. Thus, the choice of algorithm should be based on both the value s (data distribution) and Min_Sup (an analysis threshold).

6.2 Decision Plots

If we plot algorithm costs as a function of Min_Sup on the y-axis and s on the x-axis, each point in the space corresponds to a choice of algorithm. What we are really looking for is an approximate separation of the space so that in one region, the DFSVoting algorithm is faster and in the other region, the GenTest algorithm is faster. Since the decision of algorithm selection can be made simply by looking up this

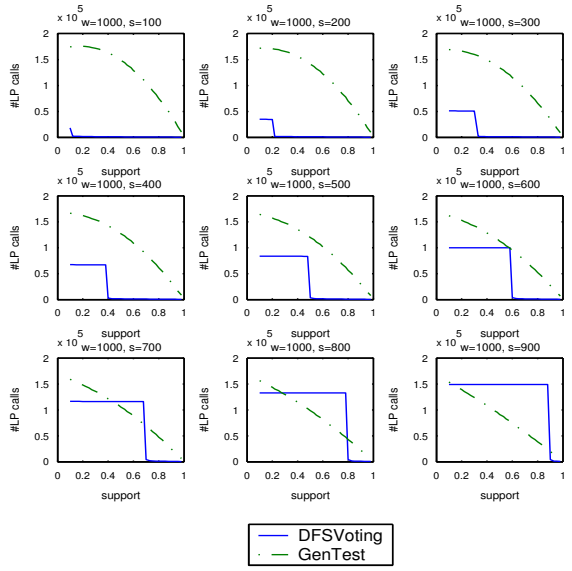

Figure 4: Algorithm Profile

precomputed information, we call such a graph a **decision plot**.

More abstractly, a decision plot for group labeling algorithms is a function $f(Min\_Sup, s)$, which takes Min_Sup and s as the input and outputs the group algorithm to use. The concept of a *decision plot* can be easily extended to deal with multiple algorithms, in which case, the whole space is divided into several regions. Each region corresponds to a particular algorithm, which is expect to perform best in that region (defined by data and analysis parameters). We can think of this extended *decision plot* as a *Voronoi diagram* [17]. Of course many other extensions can also be explored.

To choose a group labeling algorithm based on data distribution and a minimum support threshold, a *decision plot* can be derived from the *algorithm profile* shown in Figure 4. Given a fixed group size w, we use Min_Sup as the y-axis and s/w as the x-axis, and mark each point (identified by a $< Min\_Sup, s/w >$ pair) with the corresponding best algorithm, as indicated by the algorithm profiles. It gives us the graph shown in Figure 5. As we can see, the graph can be divided into two regions. The smaller triangle region corresponds to the case when GenTest is better and the other region represents the case when DFSVoting is better. It is worth noting that the boundary between regions corresponds to the intersection points in the algorithm profiles. In our particular case, the boundary of these two regions is approximately two straight lines, which suggests that we can simply fit two linear functions of Min_Sup and s to approximate the real *decision plot*. We study this approach experimentally in Section 7.

6.3 Algorithm Selection Framework

6.3.1 Estimating Data Distribution

Given a *decision plot* and a group of spectra to label, we still need a data distribution parameter s to 'lookup' the decision plot and make a choice of group labeling algorithm. Throughout the cost analysis in Section 5, we assumed that the value s is the number of identical spectra in the group.

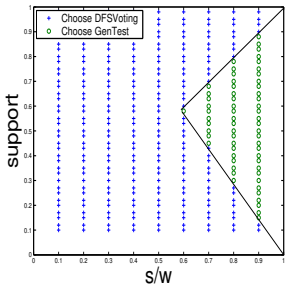

Figure 5: Decision Plot

A direct approach to estimating this value is to divide the group of spectra into clusters whose diameters are smaller than a certain threshold and use the size of the largest cluster as the value of s. Many clustering algorithms [3, 5, 8, 26] and random sampling [24] algorithms can be applied here.

6.3.2 System Graph

Now that we have discussed all the components in our algorithm selection framework, we put the pieces together in Figure 6. A given group of spectra to label first goes through the data distribution estimator, which estimates its data distribution parameter. The algorithm selector takes the estimated data distribution parameter (s), user-specified analysis parameters (e.g., Min_Sup) and looks up the decision plot to select the best algorithm. The mining engine then applies the algorithm to the input group of spectra and outputs the group label set.

In Figure 6 there are also two lines going from the output to the algorithm profile builder and data distribution estimator. This indicates that the output can serve as 'ground truth' to tune the data distribution estimator and algorithm profile. When the algorithm profile component accumulates enough data, it can in turn update the decision plot with more accurate information. In the experimental system we have built, these two feedback loops from the final output are not implemented yet. Section 7 provides more details and experimental results on the rest of the components and focuses on validating the decision plot for group labeling constructed using the theoretical cost analysis.

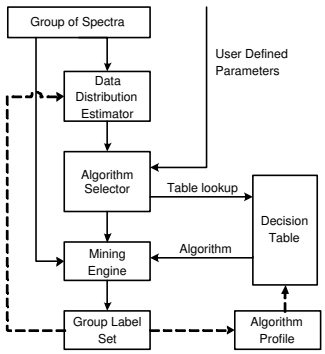

Figure 6: System Graph

7. EXPERIMENTAL RESULTS

7.1 Experimental Setting

The spectra we used in our experiments are collected from an Aerosol Time-of-Flight Mass Spectrometer. The signature database, obtained from domain experts in atmospheric aerosols, is essentially a collection of isotope distributions of chemical ions they want to detect. There are 197 signatures in the signature database, and each signature or spectrum has 255 dimensions. Notice that the performance bottleneck is not in the size of the 'signature database'. Rather, it is in the number of spectra to be labeled in a given amount of time (recall that our application involves monitoring a stream of spectra), and the cost is dominated by CPU-intensive LP calls, rather than I/O intensive disk accesses. Analogous to how a traditional DBMS seeks to minimize the cost of disk accesses, our goal is to minimize the cost of LP computation. The experimental system is implemented in C++ and runs on a 512M memory PC with Linux.

Throughout our experiments, the error bound E is set to 0.05 (a value selected heuristically after some experimentation). The threshold vector $\vec{t} = [t_1, \ldots, t_{d+1}]$ used is $\vec{t} = [0, 0.1, 0.4, 1]$. This threshold vector divides the relative quantity of a chemical element into three ranges, [0,0.1), [0.1,0.4) and [0.4,1), with each range corresponding to the state of 'missing', 'present', and 'abundant' respectively.

7.2 The Choice of Cost Metric

Throughout the cost analysis in Section 5, we used the number of LP calls as the cost metric, assuming that the number of LP calls invoked is proportional to the execution time of the algorithm. However, the time cost of a particular LP call may vary due to differences in constraints and the context of a particular LP task. To study whether the choice of LP call as a cost unit is justified, we randomly selected spectra, and recorded the number of LP calls and execution time required to label each of them. Figure 7 plots the results of our experiment, where the x-axis is the number of LP calls invoked by a particular task and the y-axis is the execution time for that task. As shown in the graph, the relation between execution time and number of LP calls invoked is clear: *The execution time is proportional to the number of LP calls invoked.*

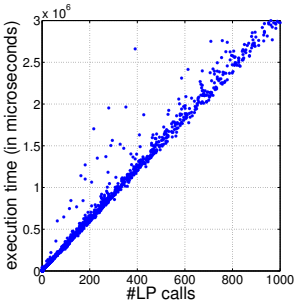

Figure 7: Number of LP calls vs.Execution Time

7.3 Algorithm Profiles and Decision Plots

For the algorithm selection framework we propose, we want to study two issues via experiments: (1) How does the algorithm profile derived from the cost estimation formula match the actual algorithm profile and how good is the cost estimation in terms of deriving the right decision plot? (2) How good is the decision plot derived from the theoretical cost model, in terms of providing the correct information for algorithm selection? In all the experimental data shown in this subsection, the group size is set to be 1000 spectra, while the error bound and threshold vector remain the same as those described in Section 7.1.

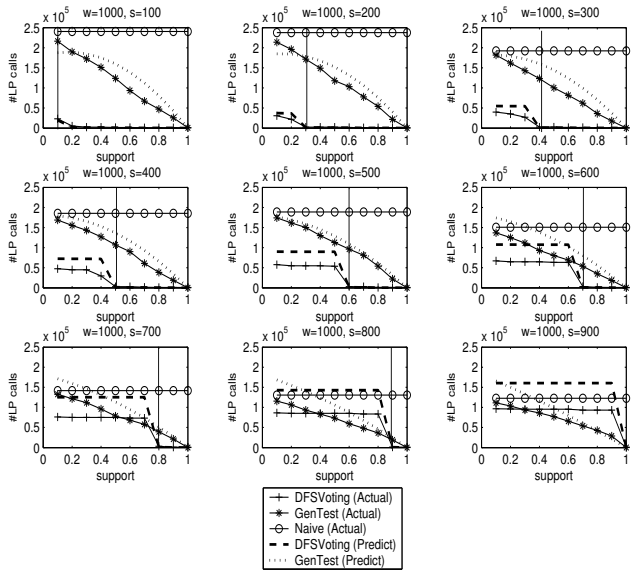

Figure 8: Experimental Result of Algorithm Profile (w=1000, n =197)

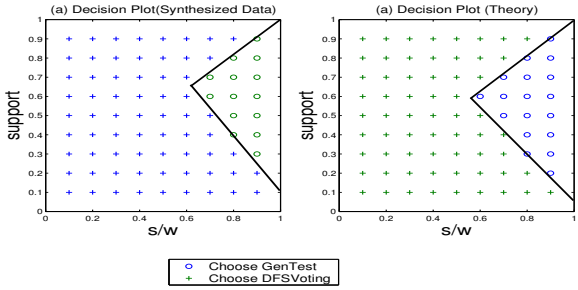

Figure 9: Experimental Result of Decision plot (w=1000, n =197)

7.3.1 Algorithm Profile

Figure 8 shows both the predicted algorithm profile and real algorithm profile for DFSVoting and GenTest. The group size w in this series of experiments is fixed at 1000 while the number of identical spectra s in the group varies from 100 to 900. Each graph shown in Figure 8 corresponds to a particular s (100, 200, ..., 900) in order from left to right and top to bottom. The series with small circles on the top of each graph shows the cost of the brute-force approach which labels all the spectra one by one.[6] The series with stars in each graph are for DFSVoting and the series with plus signs stand for GenTest. Solid lines show the real

[6]Due to the variance of average number of labels of each spectrum, the cost of brute-force approach varies from dataset to dataset

experimental results while the dotted lines are theoretical predictions plotted for comparison.

As we can see in these graphs, the theoretical prediction matches the experimental results in terms of general shape and rough absolute values. It is worth noting that both the theoretical line and experimental line of DFSVoting drop sharply around the support value of s/w, which is the point at which we have almost no group labels due to the high minimum support. While it is clear that the analytical cost estimation does not precisely predict the cost of each algorithm, it does a good job of predicting the cross-over points of the two algorithms and their relative performance, which is what we really care about for cost-based optimization: in the graphs in Figure 8 the theoretical lines cross each other at almost the same support value that the real experimental lines cross.

Going further as suggested in Section 6.2, we plot two decision plots for experimental results and theoretical prediction, respectively, in Figure 9. The left graph shows the decision plot plotted from experimental results. The right graph shows the decision table plotted from theoretical prediction. The plus signs stand for the case when the DFSVoting algorithm is better while zero signs represent the case when the GenTest algorithm is better. The two decision plots are almost exactly the same, except for four points on the lower boundary of the two regions.

For those cases where the theoretical decision plot conflicts with the real decision plot, we can see from the algorithm profile graph that the extra LP calls incurred by the wrong choice is less than 10% of the cost of the optimal algorithm. This is tolerable.

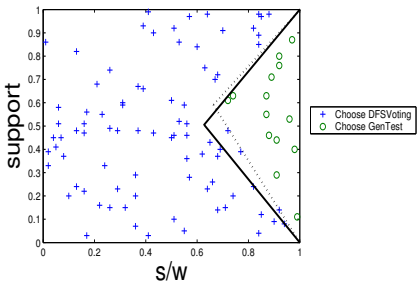

Figure 10: Decision plot for Randomly Selected Data, w=1000

7.3.2 *Decision Plots*

Figure 10 summarizes a series of experiments designed to explore the idea of performing algorithm selection by looking up the decision plot. A plus represents the case when DFSVoting is better and a zero sign stand for the case when GenTest is preferred. The solid lines separating the graph into two regions are derived from theoretical cost model. Points to the right of those lines are cases where GenTest algorithm is predicted to be faster. Points to the left of those solid lines are the cases where DFSVoting is predicted to be faster. As we can see in the graph, the solid line almost perfectly separate the plus signs and zero signs, with only a few exceptions near the borders, indicating that the decision plot derived from the theoretical cost model almost perfectly predicts the best algorithm.

7.4 Scalability

We now consider the scalability of the two proposed algorithms. We fixed the value of Min_Sup at 70%. The percentage of 'identical spectra' s/w is set to be 80%. Figure 11 shows the cost growth of each algorithm with respect to the growth of group size. Each point on the graph is the average of experimental results over 20 selected groups of spectra such that the group size w is the same for all these 20 groups. As we can see in the graph, both algorithms' costs grow linearly with respect to the group size.

Experiments with other Min_Sup and s/w values have consistently shown similar results to the one shown in Figure 11, and are omitted.

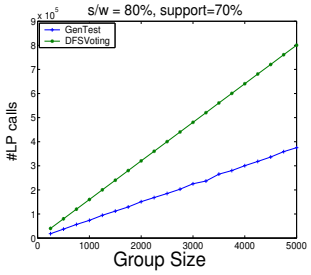

Figure 11: Scalability over group size, s/w=80%, support=70%

8. FROM MASS SPECTRA TO MASS MARKET

In previous sections, we were focused on the spectrum labeling problem. In this section, we discuss promising connections between the spectral labeling framework and market basket analysis. An obvious connection is that *after* a spectrum is labeled, we can treat it as an itemset containing the detected ions, and apply the wealth of results about itemset mining for further analysis. This is a significant benefit, since it allows us to apply powerful and widely available tools to the new problem of analyzing streams of mass spectra.

There is also a deeper and surprising connection in the other direction; we might well have a promising tool for market basket analysis in spectral labeling. In the spectrum labeling framework, we have a signature database, which represents the domain knowledge, containing profiles for chemical elements of interest. Using this, for a given spectrum we compute a label, which is essentially the most likely combination in which the known chemical ions appear in the spectrum.

If we replace chemical ion signatures by customer buying patterns that indicate underlying *phenomena* of interest, as suggested by McCarthy [16], and substitute input spectra with a customers 'market basket (purchases in a single visit to a store), then labeling offers a description of the customer by decomposing the market basket into the most plausible combination of known purchasing patterns corresponding to phenomena of interest.

For example, if we know the typical buying pattern of a doting father is a lot of toys and a few pencils, and a low income customer usually purchases a lot of chicken but very little seafood, our signature database would contain the buying patterns of these two types of customers. When a market basket containing a lot of toys, some pencils, a

lot of chicken but, no seafood is encountered, labeling will categorize that particular customer as a poor man but a doting father. In another purchase where the market basket contains a lot of toys but no food, labeling will describe the customer as a doting father, but will not be able to detect whether he is poor or rich. Such analysis was suggested as a significant direction for data mining research, called *phenomenal data mining*, in McCarthys visionary paper [16], and labeling offers promise as a tool with which to attack this intriguing application domain.

9. RELATED WORK

To our knowledge, this is the first paper to discuss labeling of groups of mass spectra, or to address cost-based data mining algorithm selection. The idea of a data mining language or framework has been explored by many researchers. In [11], Imielinski and Mannila described their vision of a data mining system, including a language specification and a general discussion of components for query compilation and execution. [12] proposed a unified algebra for multi-step data mining. [7] proposed a universal data mining model consisting of a data view, a model view and a process view. [28, 29] proposed general data mining architectures and discussed extending a DBMS with mining capabilities.

The cost analysis methodology used in this paper is similar to the analysis of the cost of index seek in [25]. An average case analysis of branch-and-bound algorithms is presented by Zhang et. al in [27]. Various aspects of numerical optimization are studied in [18]. More details on estimating the number of labels and the volume of a spectrum's solution space can be found in [14, 13, 15]. [4, 21] discuss how to describe data distributions. Clustering based techniques are surveyed in [3].

More information about spectrum labeling and environmental monitoring is provided in [10, 6, 23]. Labeled spectra are related to market baskets, to which a number of methods based on association rule mining and can be directly applied, e.g., [1, 9, 22]. Further extensions to a broader concept of phenomenal data mining is introduced in [16].

10. REFERENCES

[1] R. Agrawal et al. Mining association rules between sets of items in large databases. In *ACM SIGMOD*, 1993.

[2] A. Arning et al. A linear method for deviation detection in large databases. In *ACM KDD*, 1996.

[3] P. Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, CA, 2002.

[4] K. A. D. Peter J. Bickel. *Inference in the multiparameter case*, Chapter 6. Prentice Hall, 2 edition, 2001.

[5] C. H. Cheng et al. Entropy-based subspace clustering for mining numerical data. In *ACM KDD* , 1999.

[6] E. Gard, Jet. al. Real-time analysis of individual atmospheric aerosol particles: Design and performance of a portable atofms. In *Anal. Chem.*, pages 4083–4091, 1997.

[7] I. Geist. A framework for data mining and kdd. In *SAC*, 2002.

[8] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams. In *IEEE Symposium on Foundations of Computer Science*, 2000.

[9] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *2000 ACM SIGMOD*, 2000.

[10] Citation details omitted for anonymity

[11] T. Imielinski and H. Mannila. A database perspective on knowledge discovery. In *Comm. Of The Acm*, 39:58–64, 1996.

[12] T. Johnson et al. The 3w model and algebra for unified data mining. In *The VLDB Journal*, 2000.

[13] J. B. Lasserre. The integer hull of a convex rational polytope. In *Math. Oper. Res.*, 2003.

[14] J. B. Lasserre. A laplace transform algorithm for the volume of a convex polytope. volume 48, 2003.

[15] J. B. Lasserre and E. S. Zeron. On counting integral points in a convex rational polytope. In *Math. Oper. Res.*, 2003.

[16] J. McCarthy. Phenomenal data mining. In *Communications of the ACM 43(8)*, 2000.

[17] T. M. Mitchell. *Machine Learning*. WCB/McGraw-Hill, 1997.

[18] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 1 edition, 1999.

[19] National Research Council. *Research Priorities for Airborne Particulate Matter. Immediate Priorities and a Long-Range Research Portfolio*. 1998, National Academy Press, Washington, DC.

[20] K. A. Prather et al. *Real-time characterization of individual aerosol particles using time-of-flight mass spectrometry*. Anal. Chem., 1994; 66, 1403-1407.

[21] O. P. Rud. *Data Mining Cookbook: Modeling data for marketing, risk, and CRM*. Wiley, 1 edition, 2001.

[22] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *ACM SIGMOD*, 1996.

[23] D. Suess and K. Prather. Mass spectrometry of aerosols. In *Chemical Reviews*, pages 3007–3035, 1999.

[24] H. Toivonen. Sampling large databases for association rules. In *VLDB*, 1996.

[25] S. Yao. Approximating block accesses in database organizations. In *Communications of the ACM 20(4)*, pages 260–261, 1977.

[26] T. Zhang et al. BIRCH: an efficient data clustering method for very large databases. In *ACM SIGMOD*, 1996.

[27] W. Zhang and R. Korf. An average-case analysis of branch-and-bound with applications: Summary of results. In *AAAI*, 1992.

[28] R. Meo et al. A tightly-coupled architecture for data mining. In *ICDE*, pages 316–322, 1998.

[29] S. Sarawagi, et al. Integrating mining with relational database systems. In *ACM SIGMOD*, 1998.