Sample Selection for Maximal Diversity

Feng Pan [1], Adam Roberts [1], Leonard McMillan [1], Fernando Pardo Manuel de Villena [2],
David Threadgill [2] and Wei Wang [1]
[1]Department of Computer Science, [2]Department of Genetics
University of North Carolina at Chapel Hill
[1]{panfeng,robertsa,mcmillan,weiwang}@cs.unc.edu, [2]{fernando, dwt}@med.unc.edu

Abstract

The problem of selecting a sample subset sufficient to preserve diversity arises in many applications. One example is in the design of recombinant inbred lines (RIL) for genetic association studies. In this context, genetic diversity is measured by how many alleles are retained in the resulting inbred strains. RIL panels that are derived from more than two parental strains, such as the Collaborative Cross [2, 14], present a particular challenge with regard to which of the many existing lab mouse strains should be included in the initial breeding funnel in order to maximize allele retention. A similar problem occurs in the study of customer reviews when selecting a subset of products with a maximal diversity in reviews. Diversity in this case implies the presence of a set of products having both positive and negative ranks for each customer. In this paper, we demonstrate that selecting an optimal diversity subset is an NP-complete problem via reduction to set cover. This reduction is sufficiently tight that greedy approximations to the set cover problem directly apply to maximizing diversity. We then suggest a slightly modified subset selection problem in which an initial greedy diversity solution is used to effectively prune an exhaustive search for all diversity subsets bounded from below by a specified coverage threshold. Extensive experiments on real datasets are performed to demonstrate the effectiveness and efficiency of our approach.

1 Introduction

Selecting a sample subset sufficient to preserve the diversity seen within a given data set is a recurring problem in a variety of application domains including biology, customer review analysis and text mining. A set's diversity cover can be viewed as a variation of the classical set cover problem where at least one example including and omitting each set element is required. Furthermore, it is useful to relax the requirement of a strict cover by specifying a minimal diversity threshold (usually specified as a percentage) that is to be retained by the selected subset. The implications and motivation for finding diversity subsets also varies between application domains.

Genetic Diversity

There are many experimental scenarios where the ultimate objective is to maintain, or at least maximize, genetic diversity within relatively small breeding populations. Examples include the design of breeding programs for livestock, the captive breeding of endangered species, and the construction of recombinant inbred lines for genetic mapping in animals and plants. Allele diversity is also an important consideration when designing association studies. In the case of genetic mapping in mice, there are several existing RIL panels [15] with greater than 50 lines whose genotypes are known. Economics might dictate performing a pilot study across only a subset of the available lines [16, 10]. The following question arises: What subset preserves that greatest diversity among a set of selected markers?

Low-cost genotyping technologies provide an important tool for measuring diversity at a biomolecular level in terms of Single Nucleotide Polymorphisms (SNPs). The knowledge of a SNP's presence, frequency, and location is leveraged in a wide range of experimental designs. By definition, a SNP must be present in a minimum frequency in a population (typically 5% in human studies). We consider a SNP to be lost if it is not represented within a population sample, and our goal is to minimize this loss.

It has been previously shown in [9] that over 99% of SNPs are biallelic, which enables us to represent alleles as a binary matrix. It is straightforward to extend our approach to polymorphisms with more than two alleles.

Previously, pairwise phylogenetic distances were used to identify maximum genetic diversity subsets [7, 13]. When applied to SNPs, this approach only considers the number of inconsistencies between column pairs in the allele diversity matrix, which is less information than the full matrix that our method considers.

Besides SNPs, gene expression values in other microarray data can also be used as a measurement for genetic diversity with proper discretization. And it is a similar problem to select a subset of the samples that preserves the greatest diversity among the genes.

Customer Diversity

In e-commerce, vendors often need to solicit customer opinions on the objects (e.g., products and/or services) they provide. This feedback is valuable for profiling customers, analyzing product preferences, and building recommendation systems. There is a practical limit on the number of objects that can be listed in a questionnaire. In fact, objects should be selected carefully to maximize information for subsequent analysis. That is, they should be a small number of informative (and unbiased) representatives of all available objects of interest. Intuitively, we want the selected objects to be non-redundant and cover the full range of customer's opinions (i.e., including both positive and negative ratings). The goodness of a selection can be measured by its customer-rating diversity coverage.

A small number of selected objects is also preferred for certain data modelling tasks, such as classification. We will show in the experiment section that subsets of objects, which cover large diversity, can be used to build better (more accurate and simpler) classifiers than the full object set.

The problem of finding an optimal diversity cover is NP-Complete. An interesting variant of this problem is to find an optimal diversity subset of a given size or smaller that achieves at least a given level of diversity. In this paper, we present practical algorithms for finding such optimal diversity subsets.

Our algorithm has two phases. In the first phase, a greedy approach is used to find an initial solution and establish an achievable bound in terms of subset size and coverage. Then in the second phase, an exhaustive search for all optimal subsets is systematically performed which is seeded with parameters derived from the initial greedy solution. We then employ pruning strategies to enable an efficient search for the globally optimal solution. Extensive experiments on real datasets from three applications demonstrate the effectiveness and efficiency of our algorithm.

2 Related Work

Many algorithms have been designed to establish a summary of a data matrix such that the maximum information (or diversity) is retained.

In [11], representative elements (rows) are selected from the data matrix based on mutual information. The data matrix is considered as a joint probability distribution between the rows and columns. Rows are selected such that the maximum mutual information of the original data matrix is retained in the sub-matrix of selected representatives.

Co-clustering algorithms which cluster rows and columns of a data matrix simultaneously based on information theory are presented in [1, 5]. Dhillon *et al.* [5] generates a flat partition of the data matrix into row and column clusters that maximizes the mutual information. The algorithm proposed by Chakrabarti *et al.* [1] partitions the rows and columns such that the sum of the entropy in each cluster is minimized. The sub-matrix generated by combining the rows and columns in each cluster found by these algorithms can be considered as a summary of the original data.

Feature selection has been used extensively in the classification literature [4]. Given a class label for each row in the data matrix, the features that can maximize the classification accuracy are selected. Regardless of the diversity retention, the feature selection algorithms only select the most relevant features. These algorithms are based on heuristic methods and are not guaranteed to find an optimal solution.

The greedy solutions to the Set Cover problem and its variations are studied in [8]. These algorithms find greedy solutions which maximize the coverage over the elements at each step. Greedy solutions are only guaranteed to be within a specific ratio of the optimal solution.

3 Preliminaries

In this section, we develop notations that will be used in the paper and we provide formal problem definitions.

3.1 Diversity Cover (DC)

Let $S = \{s_1, s_2, \ldots, s_n\}$ represent a set of samples, e.g., inbred lines or objects to be reviewed, and let $M = \{m_1, m_2, \ldots, m_z\}$ represent a marker set, e.g., SNPs or reviewers. We denote the Sample-Marker matrix as H, $H = M \times S$. H is a binary matrix. In the case of SNP data, $H(i,j) = 0$ represents a majority allele at SNP m_i for sample s_j and $H(i,j) = 1$ represents that sample s_j has a minority allele at SNP m_i. While in the review data, $H(i,j) = 1$ represents that reviewer m_i ranks object s_j positively and $H(i,j) = 0$ represents that reviewer m_i ranks object s_j negatively. An example of matrix H is shown in Table 1.

Table 1. A Sample-Marker Matrix

	s_1	s_2	s_3	s_4	s_5	s_6
m_1	0	1	1	1	0	0
m_2	1	1	0	0	0	1
m_3	0	1	1	1	0	0
m_4	1	0	1	0	1	0
m_5	1	0	0	0	1	1
m_6	0	0	0	1	0	0
m_7	0	0	0	0	1	0
m_8	0	0	0	0	0	1

Given a subset of samples, $S'_j \subseteq S$, the diversity of marker m_i is covered by S'_j if and only if there are two samples in S'_j, s_l and s_k , such that $H(i,l) = 1$ and $H(i,k) = 0$. For the sample subset S'_j, the total fraction of the covered markers is called the diversity coverage of S'_j, denoted as $C(S'_j)$, $0 \leq C(S'_j) \leq 1$. For example, given the matrix in Table 1, the sample subset $\{s_4, s_5\}$ has diversity coverage 0.75, $C(\{s_4, s_5\}) = 0.75$. Obviously, for any single sample, its coverage is 0. It is generally reasonable to assume that the coverage of the entire sample set is 1, $C(S) = 1$.

Now we define the Diversity Cover problem.

Diversity Cover (DC) Problem: Given a sample set S, a marker set M and a sample-marker matrix H, find the minimum subset D, $D \subseteq S$ such that $Coverage(D) = 1$.

For example, given the sample-marker matrix in Table 1, the minimum subset that covers all the markers is $\{s_4, s_5, s_6\}$.

3.2 Diversity Cover is NP-Complete

We show DC is NP-complete via a reduction from *set cover* [3]. Given a collection of subsets, S, from the finite universal set, U, a set cover solution is the smallest group of subsets from S that covers all of U. Consider the following matrix construction for set cover. We associate each row with an element in U and each column with a subset in S. Each 1 in the matrix indicates an element's membership in a corresponding subset. Thus, set cover corresponds to finding the smallest subset of columns that provide a 1 in every row.

Suppose that we augment a set cover problem matrix with an additional row (element) and column (subset) with all 0 entries except for a single 1 at the intersection of the new row and column. This forces a DC solution to choose this newly added column. Moreover, if we ignore this added column, the remaining subsets are a solution to the original set cover problem. On the other hand, if given a solution to the original set cover problem, one can just add the last row to get a DC solution. Thus, DC is NP-complete.

3.3 Parameterized Diversity Cover (PDC)

In the Diversity Cover (DC) problem, we want to find the minimum subset that covers all markers. However, in some cases users are willing to lose the coverage of a few markers in order to find a smaller subset, e.g., in some SNP data, almost all the samples must be included to cover all the SNPs because of the large number of singleton SNPs (i.e., SNPs in which the rarer allele is present in a single strain). Therefore, we modify the DC problem by allowing a "minimum coverage ratio", ρ, rather than a full cover. Now we want to find the minimum subsets that covers no less than ρ.

Parameterized Diversity Cover (PDC) Problem: Given a sample set S, a marker set M and a sample-marker matrix H, find the minimum subset D (or subsets), $D \subseteq S$ such that $Coverage(D) \geq \rho$.

We can see that the DC problem is a special case of the PDC problem when the minimum coverage ρ is set to 1.

3.4 Upper Bound of Subset Coverage

Given a sample subset, D, $D \subseteq S$, the upper bound of its coverage can be calculated using the coverage of its subsets.

Property 3.1 *Given sample subset* $D = \{s_1, s_2, ..., s_k\}, k \geq 3$, *the upper bound of* $C(D)$ *is:*

$$C(D) \leq \frac{C(D - \{s_k\}) + C(D - \{s_{k-1}\}) + C(\{s_{k-1}, s_k\})}{2} \quad (1)$$

Proof: Let D'=$\{s_1, s_2, ..., s_{k-2}\}$, D=$D' \bigcup \{s_{k-1}\} \bigcup \{s_k\}$. Let X be the marker set covered by $D' \bigcup \{s_{k-1}\}$, Y be the marker set covered by $\{s_{k-1}\} \bigcup \{s_k\}$ and Z be the marker set covered by $D' \bigcup \{s_k\}$:

$$C(D' \bigcup \{s_{k-1}\}) = |X|/|M|, C(\{s_{k-1}\} \bigcup \{s_k\}) = |Y|/|M|,$$

$$C(D' \bigcup \{s_k\}) = |Z|/|M|.$$

Let W be the marker set covered by D . We have

$$W = X \bigcup Y \bigcup Z$$

For any marker m_l in Z, either it is already covered by D' or it is only covered when sample s_k is considered together with D'. In the first case, m_l also belongs to X since markers in X are covered by $D' \bigcup \{s_{k-1}\}$. In the second case, all the samples in D' have the same value on marker m_l and sample s_k has the opposite value on m_l. If sample s_{k-1} has the same value on m_l as s_k, m_l is also covered by $D' \bigcup \{s_{k-1}\}$ and belongs to X. Or if s_{k-1} has the opposite value on m_l compared to s_k, m_l is covered by $\{s_{k-1}\} \bigcup \{s_k\}$ and belongs to Y. Therefore we have

$$Z \subseteq X \bigcup Y \quad (2)$$

$$W = X \bigcup Y$$

$$C(D) = (|X| + |Y| - |X \bigcap Y|)/|M| \quad (3)$$

For any marker m_l in $X - Y$ which is covered by $D' \bigcup \{s_{k-1}\}$ but not by $\{s_{k-1}, s_k\}$, we know that m_l is either covered by D' alone or by D' together with $\{s_{k-1}\}$ and we know that s_k has the same value as s_{k-1} on m_l. Therefore, in either case, m_l is also covered by $D' \bigcup \{s_k\}$ and belongs to Z. Similarly, for any marker m_l in $Y - X$ which is covered by $\{s_{k-1}, s_k\}$ but not $D' \bigcup \{s_{k-1}\}$, we know that samples in D' have the same value as s_{k-1} on m_l while sample s_k has the opposite value to s_{k-1} on m_l. Therefore, m_l is also covered by $D' \bigcup \{s_k\}$ and belongs to Z. We can get

$$X - Y \subseteq Z, Y - X \subseteq Z \quad (4)$$

Since $(X - Y) \bigcap (Y - X) = \emptyset$, we have

$$|Z| \geq |X| - |X \bigcap Y| + |Y| - |X \bigcap Y| \quad (5)$$

According to Equations 3 and 5,

$$C(D) = \frac{|X| + |Y| - |X \bigcap Y|}{|M|} \leq \frac{|X| + |Y| + |Z|}{2|M|}$$

Therefore,

$$C(D) \leq \frac{C(D - \{s_k\}) + C(D - \{s_{k-1}\}) + C(\{s_{k-1}, s_k\})}{2}$$

□

When the subset D contains only 3 samples, the upper bound in Equation 1 becomes the exact value of $C(D)$.

Property 3.2 *Given the pair-wise diversity coverage of three samples,* s_i, s_j *and* s_k, *the coverage of set* $\{s_i, s_j, s_k\}$ *is known.*

$$C(\{s_i, s_j, s_k\}) = \frac{C(\{s_i, s_j\}) + C(\{s_i, s_k\}) + C(\{s_j, s_k\})}{2} \quad (6)$$

The proof is similar and is omitted for brevity. Obviously, by using Equations 1 and 6 recursively, we can establish an upper bound of subset coverage using only the pair-wise coverages.

Theorem 3.1 *Given a sample subset* $D = \{s_1, s_2, ..., s_k\}$, *we can calculate the upper bound of* $C(D)$ *using only the pair-wise coverage* $C(\{s_i, s_j\})$, $s_i, s_j \in D$ *according to Equations 1 and 6.*

For example, the upper bound of $C(\{s_1, s_2, s_3, s_4\})$ is
$C(\{s_1, s_2, s_3, s_4\}) \leq [2C(\{s_1, s_2\}) + 2C(\{s_3, s_4\})$
$+C(\{s_1, s_3\}) + C(\{s_2, s_3\}) + C(\{s_1, s_4\}) + C(\{s_2, s_4\})]/4$
Note that for a sample subset D, we can get several coverage upper bounds based on Theorem 3.1 by exchanging the order of the samples in D. We discuss the details of calculating a diversity upper bound using Theorem 3.1 in Section 4.2.4.

4 Algorithms

In this section, we present our Exhaustive Subset Enumeration (*ESE*) algorithm that solves the Parameterized Diversity Problem. Our algorithm guarantees to find all the minimum sample subsets that have diversity coverage no less than ρ. The *ESE* algorithm has two phases. In the first phase, a greedy algorithm, Parameterized Greedy Diversity Subset (*PGDS*), is used to find an initial sample subset S_G that has $C(S_G) \geq \rho$. Then in the second phase, we present an optimal K-ρ Diversity Subset ($K\rho DS$) algorithm to exhaustively search for all sample subsets with sizes K and smaller and with coverages no less than ρ. The initial sample subset S_G and several pruning strategies are used to reduce the searching space. The pseudocode of the *ESE* algorithm is shown in Figure 1.

Input:
- Sample Set S, Marker Set M, Sample-Marker Matrix H
- Minimum Diversity Coverage ρ

Output: A set of minimum sample subsets, I. $\forall S' \in I, C(S') \geq \rho$ **Method:**

1. S_G=*PGDS*(ρ, S, M, H).
2. K=$|S_G|$.
3. I=$K\rho DS$(K,ρ, S, M, H).

Figure 1. The *ESE* Algorithm

4.1 Parameterized Greedy Diversity Subset Algorithm

In Section 3, we proved that Diversity Cover is NP-complete and can be mapped to Set Cover. There is a well known greedy algorithm for the Set Cover problem. It chooses the subset that maximizes the increase in coverage in each step until all the elements are covered. The greedy algorithm can achieve an approximation ratio of $H(z)$, $H(z) = \sum_{k=1}^{z} \frac{1}{k} \leq \ln z + 1$ and $z = |M|$ [3].

In the first phase of *ESE*, we design a similar algorithm, Parameterized Greedy Diversity Subset (*PGDS*), to find greedy approximations to the Parameterized Diversity Cover problem. The *PGDS* algorithm also chooses the sample that maximizes the increase in the diversity coverage in each step. There are two differences in the *PGDS* algorithm compared with the greedy approach of the Set Cover problem.

1. The *PGDS* algorithm cannot pick the best first sample based on coverage because every single sample has zero coverage. Therefore, *PGDS* is restarted with each sample and we pick the smallest subset from the n generated subsets, where n is the number of samples.
2. The *PGDS* algorithm stops once the coverage of the sample subset exceeds the minimum threshold ρ.

The details of the *PGDS* algorithm are shown in Figure 2. The algorithm considers each sample in step 2, and the minimum subset among all the S' is reported as S_G. The time complexity of *PGDS* is $O(kn^2)$ where $k = |S_G|$.

Input:
- Sample Set S, Marker Set M, Sample-Marker Matrix H
- Minimum Coverage ρ

Output: sample subset S_G having $C(S') \geq \rho$
Method:

1. $S_G = \{\}$.
2. for all $s_i \in S, i = 1 \ldots n$.
3. $S' = \{s_i\}, R = S - S', c = 0$.
4. while $c < \rho$
5. $S' = S' \bigcup \{s_l\}$, which
   $C(S' \bigcup \{s_l\}) = Max_{s_j \in R}(C(S' \bigcup \{s_j\}))$.
6. $R = R - \{s_l\}, c = C(S')$.
7. if $|S'| < |S_G|$
8. $S_G = S'$.

Figure 2. The $PGDS$ Algorithm

For example, if we are given the matrix in Table 1 and set $\rho = 1$, the subset found by *PGDS* is $S_G = \{s_1, s_4, s_5, s_6\}$ which is larger than the optimal minimum subset $\{s_4, s_5, s_6\}$.

4.2 Optimal K-ρ Diversity Subset Algorithm

In the first phase of the *ESE* algorithm, the *PGDS* algorithm finds an initial subset S_G satisfying $C(S_G) \geq \rho$. It establishes an upper bound on the size of the optimal subsets in I, i.e., any subset S' which has $C(S') \geq \rho$ should have size smaller than or equal to subset S_G. Let K=$|S_G|$, the exhaustive enumeration need to be performed only on the subsets having size no larger than K. The exhaustive enumeration can take exponential time in principle. However, with efficient pruning strategies, our enumeration algorithm, $K\rho DS$, performs much better in practice, finding the optimal subsets quickly.

The $K\rho DS$ algorithm searches all possible combinations of samples up to size K in an enumeration tree. Figure 3 illustrates part of the enumeration tree of the matrix in Table 1 and represents our search when we do not apply any pruning strategies. Each node in the tree stores a sample subset S' and the corresponding $C(S')$. The root represents the empty set. For each child node, the sample subset has one more sample than its parent node.

The $K\rho DS$ algorithm performs a depth-first search [3] on the enumeration tree. By imposing an order on the samples, the algorithm is able to perform a systematic search by enumerating all combinations, i.e., no combination is missed or revisited. Without loss of generality, let's assume the order is $s_1, s_2, \ldots, s_n$. For example, the depth-first search order on the enumeration tree in Figure 3 is

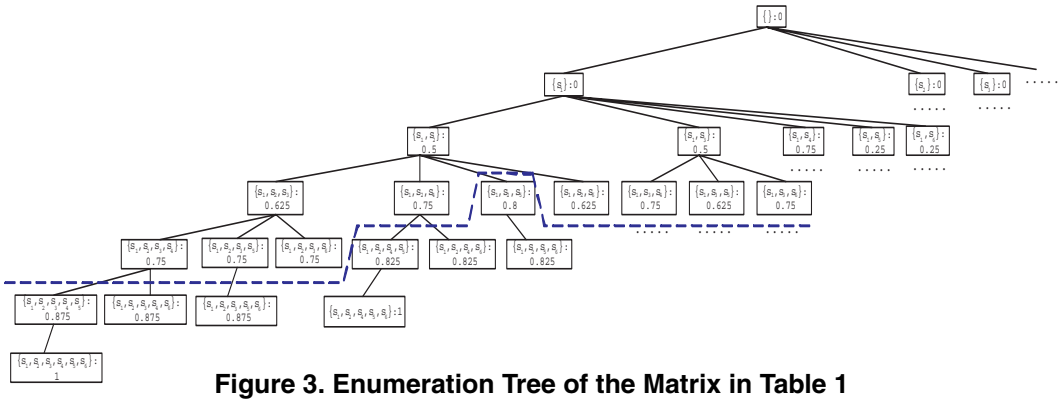

Figure 3. Enumeration Tree of the Matrix in Table 1

$\{s_1, s_1s_2, s_1s_2s_3, s_1s_2s_3s_4, s_1s_2s_3s_4s_5, s_1s_2s_3s_4s_5s_6, s_1s_2s_3s_4s_6, s_1s_2s_3s_5, s_1s_2s_3s_5s_6, s_1s_2s_3s_6, \ldots\}$.

Among all the subsets that achieve the coverage threshold ρ, only the minimum sample subsets are reported. For example, if we set $\rho = 0.8$, only the nodes $\{s_1, s_2, s_3\}$ and $\{s_4, s_5, s_6\}$, which are below the dashed line in Figure 3, satisfy the ρ-threshold and subset-size constraints. Note that among all the nodes below the line, only those on the boundary need to be examined since nodes below the boundary have subsets of larger size. Details of the basic $K\rho DS$ algorithm are shown in Figure 4.

Input:

- Sample Set S, Marker Set M, Sample-Marker Matrix H
- Minimum Coverage ρ, Maximum Size K, $K = |S_G|$

Output: A set of minimum sample subset, I,

$$\forall S' \in I, C(S') \geq \rho, |S'| \leq K$$

Method:

1. *Initialize*.
   - Candidate minimum sample subset list, *cList*=$\emptyset$.
   - Current sample subset, *cSample*=$\emptyset$.
   - Remaining sample subset, *rSample*=S.
2. Enumerate(*cSample*, *rSample*).
3. I=the minimum subsets in *cList*.

Subroutine: Enumerate(*cSample'*, *rSample'*)
Method:

1. if $|$*cSample'*$| \geq K$
2. return.
3. for each $s_i \in$ *rSample'*
4. if *C*(*cSample'* $\bigcup \{s_i\}) \geq \rho$
5. Insert set *cSample'* $\bigcup \{s_i\}$ into *cList*.
6. else
7. *cSample"*=*cSample'* $\bigcup \{s_i\}$.
8. *rSample"*=*rSample'* - $\{s_i\}$.
9. Enumerate(*cSample"*, *rSample"*).
10. *rSample'*=*rSample'* - $\{s_i\}$.

Figure 4. The $K\rho DS$ Algorithm

The enumeration tree is dynamically materialized according to a depth-first searching order. At each node, the coverage of the corresponding sample subset S' is calculated based on the sub-matrix $H' = M' \times S'$, where M' is the set of markers that are not covered by the sample set in the parent node. The use of the dynamically generated sub-matrix can efficiently reduce the runtime of the $K\rho DS$ algorithm.

In the worst case, the $K\rho DS$ algorithm takes exponential time. In order to accelerate the search, we use several pruning strategies to reduce the search space.

4.2.1 Pruning Strategy 1: Dynamically Limit the Size of the Minimum Sample Subset

In the first phase of *ESE*, the greedy algorithm *PGDS* provides an upper bound of the size of the minimum sample subset. When the $K\rho DS$ algorithm searches the enumeration tree, it does not need to check any node of more than K samples.

The size of the minimum sample subsets can also be updated dynamically during enumeration. It is possible that K may be larger than the minimum size. The value of K is updated to be the size of the smallest subset S', satisfying $C(S') \geq \rho$, found so far. All remaining nodes representing larger subsets can be pruned from the enumeration tree without further examination. For example, if K is 9 and the algorithm finds a subset of 8 samples that can satisfy the threshold ρ, K is revised to 8 and any subsequent subsets of more than 8 samples are pruned from the enumeration tree.

Pruning strategy 1 is applied at step 5 of subroutine **Enumerate()** in Figure 4. When *cSample'* $\bigcup \{s_i\}$ is inserted into *cList*, its size is compared with that of the smallest subset in *cList*. If *cSample'* $\bigcup \{s_i\}$ is smaller, K can be updated accordingly and all the subsets in *cList* having larger size can be dumped.

4.2.2 Pruning Strategy 2: Order Samples by Pair-wise Coverage

For each node, we can estimate the increase in coverage for each sample from $rSample'$ based on its pair-wise coverage with every sample in $cSample'$. For example, in Figure 3, consider node $cSample' = \{s_1, s_2\}$, $rSample' = \{s_3, s_4, s_5, s_6\}$. We know that the pair-wise coverages are:

- $\mathbf{s_3}$: $C(\{s_1, s_3\}) = 0.5, C(\{s_2, s_3\}) = 0.25$
- $\mathbf{s_4}$: $C(\{s_1, s_4\}) = 0.75, C(\{s_2, s_4\}) = 0.25$
- $\mathbf{s_5}$: $C(\{s_1, s_5\}) = 0.25, C(\{s_2, s_5\}) = 0.75$
- $\mathbf{s_6}$: $C(\{s_1, s_6\}) = 0.25, C(\{s_2, s_6\}) = 0.5$

For each sample in $rSample'$, we use the sum of its pair-wise coverage with each sample in $cSample'$ as its score. This score is an (optimal) estimate of the additional coverage this sample can bring.

$Score(s_3) = 0.75, Score(s_4) = 1, Score(s_5) = 1, Score(s_6) = 0.75$

We sort, in descending order, the samples in $rSample'$ based on their scores so that in the sub-tree of node $cSample' = \{s_1, s_2\}$, sample s_4 and s_5 will be added first followed by s_3 and s_6. We can see that subsets having larger coverage are searched first in this case.

The sample sorting is conducted at each node dynamically. The pair-wise coverage of all samples can be calculated in advance and retrieved to compute the scores. At the root of the enumeration tree, the samples are initially sorted according to their order selected by the *PGDS* algorithm.

Pruning strategy 2 can be used before step 3 of subroutine **Enumerate()** in Figure 4. Samples in *rSample'* are sorted accordingly.

In some cases where the estimated size of minimum subsets, K, by *PGDS* is equal to or close to their actual size, pruning strategy 2 itself cannot reduce the search space dramatically. However, when combined with the following pruning strategy, it always delivers a substantial improvement in efficiency.

4.2.3 Pruning Strategy 3: Estimate a Branch Upper Bound on Coverage

The coverage of sample subsets generally increases monotonically when adding new samples. For each node in the enumeration tree, we can calculate an upper bound, $C(cSample' \bigcup rSample')$, on the coverage of any sample subsets represented in the subtree. Any subsets represented in the branch must have coverage no larger than that value. We call it the *branch-upper-bound*. For example, consider node $\{s_1, s_3, s_5\}$ in Figure 3, $cSample' = \{s_1, s_3, s_5\}$ and $rSample' = \{s_6\}$. The upper bound is $C(\{s_1, s_3, s_5, s_6\})$, which is 0.825. If the *branch-upper-bound* of the subtree is less than the minimum coverage threshold ρ, we can safely prune the subtree.

It is inefficient to calculate the upper bound at each node independently by adding up the samples in $cSample'$ and $rSample'$. However, we can calculate it simultaneously with the depth-first search by tracking the samples that are absent in the subtree under each node.

Given a node with its $cSample'$ and $rSample' = \{s_{i_1}, s_{i_2}, ..., s_{i_q}\}$, its left-most child node has the same *branch-upper-bound*. Let $cSample''_1$ and $rSample''_1$ be the current and remaining samples at the left-most child node. We have

$$cSample''_1 \bigcup rSample''_1 = (cSample' \bigcup \{s_{i_1}\}) \bigcup (rSample' - \{s_{i_1}\}) = cSample' \bigcup rSample'$$

where s_{i_1} is the first sample in $rSample'$.

For the j^{th} child nodes ($1 < j \leq q$) of the current node, we have

$$cSample''_j \bigcup rSample''_j = (cSample''_{j-1} \bigcup rSample''_{j-1}) - \{s_{i_{j-1}}\}$$

Therefore, we can calculate the *branch-upper-bound* of a node according to the upper bound of its parent node or its siblings. The *branch-upper-bound* at the root node is 1 since every sample appears in some nodes. When we proceed along a branch, this value decreases as more samples are absent in the sub-tree. For example, if we know that the upper bound at node $\{s_1, s_3\}$ in Figure 3 is 1,

- for its child node $\{s_1, s_3, s_4\}$, the upper bound is still 1 because $\{s_1, s_3, s_4\}$ is the left-most child node of $\{s_1, s_3\}$.
- for $\{s_1, s_3, s_5\}$, sample s_4 is absent, its upper bound becomes 0.875.
- for $\{s_1, s_3, s_6\}$, sample s_4 and s_5 are absent. Its branch upper bound on coverage becomes 0.75.

Pruning strategy 3 can be used before step 4 of subroutine **Enumerate()** in Figure 4. If the upper bound on branch coverage is less than ρ, the subroutine can stop and return to its previous level.

As mentioned earlier, pruning strategy 2 can improve the efficiency of pruning strategy 3. After we sort the succeeding samples at each node in the tree, the last several branches are likely to be pruned by strategy 3 because they contain only those samples that have the least increase in coverage. Our experiments on real datasets suggest that using pruning strategies 1 and 3 together reduces the runtime of the $K\rho DS$ algorithm by $70\% - 80\%$. Combining pruning strategies 1, 2 and 3 can reduce the runtime by more than 95%.

4.2.4 Pruning Strategy 4: Refine the Branch Upper Bound on Coverage

In pruning strategy 3, we estimate the *branch-upper-bound* using the current sample subset and all its succeeding samples in *rSample'*. This upper bound is loose because in many cases, we cannot include all the succeeding samples into the current subset. For example, if the current node represents a subset of p samples and there are q succeeding samples in *rSample'*, we can at most include a subset of $K - p$ samples from *rSample'* during the search in the subtree under the current node. If we can calculate the maximum increase in coverage after adding any subset of $K - p$ samples from *rSample'*, we get a tighter upper bound than the one in pruning strategy 3.

Suppose that the current sample subset is $cSample = \{s_{i_1}, s_{i_2}, ..., s_{i_p}\}$ and the succeeding samples are $rSample = \{s_{j_1}, s_{j_2}, ..., s_{j_q}\}$. $cSample$ covers the marker subset M_a, $M_a \subset M$, and the uncovered marker subset is M_b, $M_b = M - M_a$. Since marker set M_b is uncovered by $cSample$, all the samples in $cSample$ have the same value on each marker in M_b. Therefore, we can use a dummy sample s_{j_0} to represent the diversity of $cSample$ on M_b. When adding a subset of samples from $rSample$, S', into the current subset $cSample$, the increase of coverage is the coverage of $S' \bigcup \{s_{j_0}\}$ on M_b. We can calculate the pair-wise coverage on M_b between any two samples in $\{s_{j_0}, s_{j_1}, s_{j_2}, ..., s_{j_q}\}$. Let the set of pair-wise coverage be $C_{pair} = \{c_1, c_2, ..., c_m | m = (q+1)q/2\}$. Note that coverage is still calculated based on $|M|$, so that the total coverage on M can be calculated by adding the coverage on M_b (increase of coverage) and the coverage on M_a (current coverage) together.

In order to know the maximum increase in coverage after adding any subset of $K - p$ samples from *rSample'*, we need to calculate the upper bound of the coverage of any

$(K-p)$ samples from $rSample$ together with s_{j_0} on M_b. However, in order to make the problem easier, we loosen the requirement and calculate the upper bound of the coverage of any $K-p+1$ samples of *rSample'*$\bigcup\{s_{j_0}\}$ on M_b.

According to Theorem 3.1 in Section 3, by recursively applying Equation 1, we can get the upper bound of the coverage of any u samples using their pair-wise coverage. The upper bound should be in the following form

$$C_{max} \leq \sum_{i=1}^{(u-1)u/2} a_i \cdot C(s_j, s_k), a_1 \geq a_2 \geq a_3 ... \quad (7)$$

Now we have $q+1$ samples in total and we know the set of all their pair-wise coverage C_{pair}, we can calculate the upper bound of the coverage of any subset of $K-p+1$ samples by replacing $C(s_j, s_k)$ in Equation 7 with the $\frac{(K-p)(K-p+1)}{2}$ largest pair-wise coverage in C_{pair}. If the pair-wise coverage in C_{pair} are sorted in descending order, the upper bound of increasing coverage after adding $K-p$ samples into $cSample$ is

$$\triangle C \leq \sum_{i=1}^{(K-p)(K-p+1)/2} a_i \cdot c_i, c_i \in C_{pair} \quad (8)$$

Let C be the current coverage, $C = \frac{|M_a|}{|M|}$. If $C + \triangle C$ is still less than ρ, the $K\rho DS$ algorithm does not need to search the subtree under the current node because there is no sample subset in the subtree that is not larger than K in size and with coverage not less than ρ.

Equation 8 provides a tighter upper bound than the one in pruning strategy 3 especially when there are large number of samples in the data. However, in order to get the upper bound, pair-wise coverage on M_b between $\{s_{j_0}, s_{j_1}, s_{j_2}, ..., s_{j_q}\}$ must be computed. Note that the coefficients a_i in Equations 7 and 8 are constants and can be calculated for each size of sample sets in advance. The computation and the pruning can be inserted before step 7 of subroutine **Enumerate()** in Figure 4. We can see that pruning strategies 3 and 4 are used in different places in the algorithm. In fact, these two strategies can be used together though strategy 4 provides tighter upper bound. Pruning strategy 3 is much faster than strategy 4 and, therefore, it is used as the pre-pruning step before pruning with strategy 4.

Though calculating the pair-wise coverage of $\{s_{j_0}, s_{j_1}, ..., s_{j_q}\}$ at each node takes time, we demonstrate in our experiments that the extra time used for coverage calculation is negligible compared with the runtime saved by pruning branches using pruning strategy 4. Also, as a side product, the actual increase in coverage for adding each sample from $rSample'$ into $cSample'$ is known during the pair-wise coverage calculation. Therefore, in pruning strategy 2, instead of ordering the samples by the estimated score, the algorithm can now order the samples in $rSample'$ by their actual increase in coverage.

5 Experiments

In this section, we present results on synthetic and real data to show the efficiency of our algorithms and the effectiveness of the selected sample subsets. One real dataset is a SNP panel from recombinant inbred mouse strains. The other two real datasets are of customer review type.

Data

- Perlegen data[1]: The Perlegen dataset contains genotypes from 15 commonly used laboratory mouse strains[2], {129S1/SvImJ, A/J, AKR/J, BALB/cByJ, BTBR T+ tf/J, C3H/HeJ, CAST/EiJ, DBA/2J, FVB/NJ, KK/HlJ, MOLF/EiJ, NOD/LtJ, NZW/LacJ, PWD/PhJ, WSB/EiJ} and a reference strain (C57BL/6J). These 16 strains account for over 85% of all inbred strains used in biomedical research. The dataset contains $8,322,543$ SNPs in total. The dataset is imputed using the method described in [12].

- Congressional Voting Records data[3]: The voting dataset includes votes from 435 Congressmen on 16 key votes. The votes can be 'yes' or 'no' and are denoted by 1 and 0. The 435 congressmen are classified into two groups, 267 democrats and 168 republicans.

- Jester data[4][6]: The Jester dataset contains 4.1 Million ratings (-10.00 to +10.00) of 100 jokes from 73,421 users. We discretize the data by replacing positive ratings by 1 and negative ratings by 0. Since none of the 73,421 users completes the review for all the 100 jokes, we use jokes that were reviewed by more than 70% of the users and then select users who reviewed all these jokes. Thus, the dataset we use contains 46,268 users and 30 jokes without missing values.

- Synthetic data: The synthetic data is randomly generated. The dataset is a binary matrix consisting of $40,000$ rows and 100 columns. We consider the rows as markers and columns as samples.

The synthetic dataset is mainly used to demonstrate the efficiency of our algorithms. And the three real datasets are mainly used to demonstrate the effectiveness of the selected sample subsets.

Except as otherwise noted, we use all the four pruning strategies collectively in the experiments because this combination provides the best runtime performance. The algorithms are implemented in MATLAB, and all experiments are conducted on a PC with CPU P4 3GHz, 1G RAM and 80G HDD.

5.1 Efficiency Analysis

In this section, we demonstrate the efficiency of the *PGDS* and the $K\rho DS$ algorithm using the synthetic data and some of the real datasets.

[1] http://mouse.perlegen.com/mouse/index.html
[2] We regard each mouse strain as a sample.
[3] http://www.ics.uci.edu/ mlearn/MLSummary.html
[4] http://www.ieor.berkeley.edu/ goldberg/jester-data/

Scalability

For the synthetic data, we vary the number of rows, the number of columns and the minimum coverage ρ respectively. The default values for these settings are: number of rows=$40k$, number of columns=80 and minimum coverage ρ=0.965. While we are varying one of the settings, the other two use the default values. The runtime performance of *PGDS* and $K\rho DS$ is shown in Figure 5. The runtime of both algorithms increases linearly when the number of rows increases in Figure 5(a). And the runtime increases quadratically when the number of columns and ρ increase for both algorithms as shown in Figure 5(b) and (c).

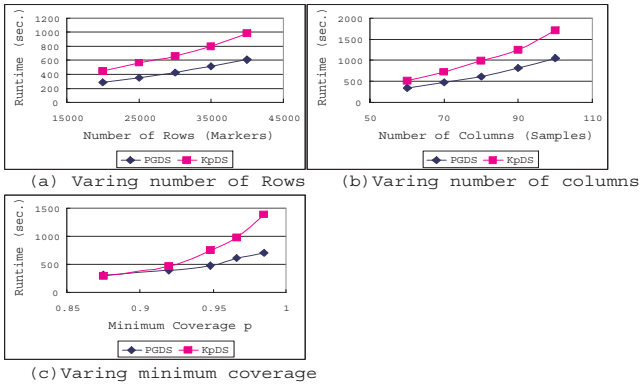

Figure 5. Scalability: Runtime on Synthetic Data

For the real datasets, we only vary the minimum coverage ρ setting and use all the rows and columns. Both *PGDS* and $K\rho DS$ can finish searching the Voting data in 1 second. Therefore, we only show the runtime performance on Perlegen and Jester data in Figure 6.

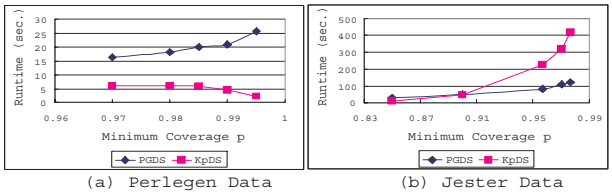

Figure 6. Scalability: Runtime on Real Data

The runtime performance on the Jester data is similar to that of the synthetic data for both algorithms. For Perlegen data, $K\rho DS$ can even be faster than *PGDS* because of the pruning strategies. Also, the runtime of $K\rho DS$ begins to drop when ρ is larger than 0.985. The reason is that the entire searching space becomes smaller when $\rho > 0.985$. The minimum subsets have size larger than 9 when $\rho > 0.985$ and causes the shrinking of the entire searching space because a subset of 10 samples or more contains more than half of the samples in the data.

Note that the total runtime of the *ESE* algorithm is the sum of the runtime of *PGDS* and $K\rho DS$.

Comparison of Subsets Found by *PGDS* and $K\rho DS$

As we discussed in Section 4, the sample subsets found by *PGDS* may not be the optimal subset, i.e., either there exists a smaller subset that can achieve the minimum coverage ρ or there exists a subset with the same size but has larger coverage. Thus, in this section, we compare the subsets found by *PGDS* and $K\rho DS$. In the first part of the experiments, we vary minimum coverage ρ and compare the size of the minimum subsets found by both algorithms. Then we use the set of sizes of the minimum subsets found by $K\rho DS$ in the first part, and compare the optimal coverage that is achieved by the two algorithms for each of the subset size. The results are shown in Figure 7.

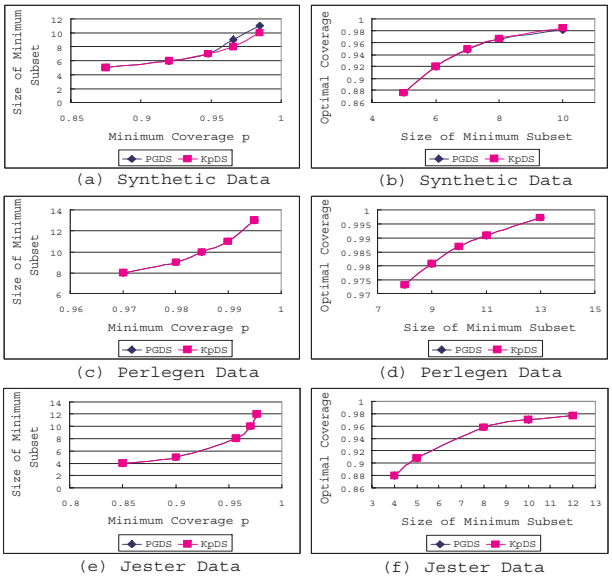

Figure 7. Comparison of Subsets

As we can see, on the synthetic dataset, *PGDS* finds a larger subset when ρ becomes large. And for subsets of size 8 and 10, the true optimal subset found by $K\rho DS$ has larger coverage than the subset found by *PGDS*. However, as shown in Figure 7(c-f), *PGDS* always find the same optimal subset as $K\rho DS$ on the real datasets. The result on the Voting data is the same as the Perlegen and Jester data and is omitted.

Efficiency of Pruning Strategies

In this section, we compare the efficiency of the pruning strategies discussed in Section 4. We vary the minimum coverage parameter ρ and compare the runtime performance on Perlegen and Jester data. The synthetic data is not used because it is too large for $K\rho DS$ to search without any one of the pruning strategies. And the Voting data is too small to be used to show the difference.

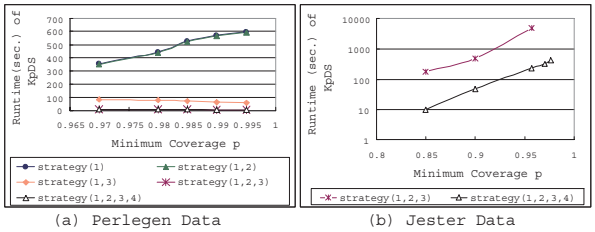

Figure 8. Efficiency of Pruning Strategies

Figure 8(a) shows the results on the Perlegen data. We observe that pruning strategies (1,2,3) and pruning strategies (1,2,3,4) give the best performance, which is orders of magnitude faster than other strategy combinations. The reason that pruning strategy 4 does not improve the performance significantly when used in combination with strategies 1,2 and 3 is that the Perlegen dataset only contains 16 samples. This is sufficiently small that the two upper bounds from strategies 3 and 4 are close to each other. Using only pruning strategies 1 and 2 (sorting) just slightly reduces the runtime. This is because sorting only helps the $K\rho DS$ algorithm find minimum subsets faster, but cannot reduce the search space by pruning sub-trees. Using only pruning strategies 1 and 3 saves about $70\% - 80\%$ of the runtime of enumerating with strategy 1 only. Without sorting, strain subsets offering good coverage are randomly distributed in the enumeration tree. Strain sorting helps to bring these branches together in the enumeration tree so that effective pruning can be achieved.

On the Jester data, the $K\rho DS$ algorithm can finish the tasks in reasonable time only with pruning strategies (1,2,3) or pruning strategies (1,2,3,4). And for pruning strategies (1,2,3), it also can not afford a minimum coverage ρ larger than 0.96. As we can see, pruning strategies (1,2,3,4) are orders of magnitude faster than pruning strategies (1,2,3). As we discussed in Section 4, pruning strategy 4 has a large advantage over pruning strategy 3 when the number of samples becomes large.

5.2 Effectiveness Analysis

In this section, we apply our algorithm on the three real datasets and demonstrate the effectiveness by analyzing the selected sample subsets.

Perlegen Data

The Perlegen dataset has 16 samples and more than $8M$ SNPs. As we discussed in Section 1, in the design of recombinant inbred lines, an important measurement for a set of lines (samples) is its diversity coverage on the SNPs. A subset of 8 strains was hand selected for the Collaborative Cross [2] by biologists based on the phylogenetic relationships assumed for strains. We compare this subset with the best 8-strain subsets found by the *ESE* algorithm in Table 2.

Table 2. Perlegen Data: Comparing the 8-strain subsets of the Collaborative Cross with the maximum diversity solution found by *ESE*

		$Coverage$
Collaborative Cross Subset	**129S1/SvImJ**, **CAST/EiJ**, **PWD/PhJ**, **WSB/EiJ**, NZW/LacJ, C57BL/6J, NOD/LtJ, A/J	0.8926
ESE Subset	**129S1/SvImJ**, **CAST/EiJ**, **PWD/PhJ**, **WSB/EiJ**, KK/HlJ, DBA/2J, MOLF/EiJ, FVB/NJ	0.9575

As we can see, the *ESE* subset achieves higher coverage than the Collaborative Cross subset. Four strains are common to both subsets: **129S1/SvImJ**, **CAST/EiJ**, **PWD/PhJ** and **WSB/EiJ**. Aside from **129S1/SvImJ**, all strains are wild-derived from the three major Mus musculus subspecies. We plot the distribution of the diversity coverage of random set of 8 samples in Figure 9. The coverage of the Collaborative Cross subset, 0.8926, is labelled by the red dotted line in the figure. The Collaborative Cross subset has coverage larger than more than 70% of the randomly selected 8-sample subsets while the *ESE* subset is obviously the one has the largest coverage.

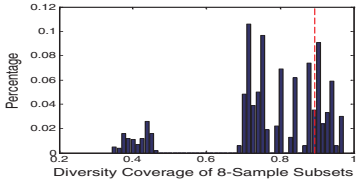

Figure 9. Perlegen Data: Distribution of Diversity Coverage of 8-sample Subsets

Voting Data

The Voting data includes votes of the 435 Congressmen on 16 key votes. We consider the congressmen as markers and the key votes as samples. Our *ESE* algorithm finds two sample subsets that consist of 5 samples and have diversity coverage $= 1$, i.e., all the markers (congressmen) are covered. The two subsets are listed in Table 3.

Table 3. Voting Data: Subsets of 5 Samples found by *ESE* that have coverage $= 1$

Subset 1	handicapped-infants, physician-fee-freeze religious-groups-in-school, mx-missile, duty-free-exports
Subset 2	physician-fee-freeze, el-salvador-aid anti-satellite-test-ban, mx-missile, synfuels-corporation-cutback

As we discussed in Section 1, these subsets can be used to generate simpler yet more accurate classification models. We use Weka [5], which is a data mining software in Java, to build different classifiers based on all the 16 samples and the two 5-sample subsets. Classification accuracy is calculated by using 10-fold cross-validation. The accuracy of the classifiers are listed in Table 4.

Table 4. Voting Data: Accuracy of Classifiers based on full set and subsets

Classifiers	Full Sample Set	Subset 1	Subset 2	Random Subset
RandomTree	92.8%	**95.4**%	95.17%	86.66%
PART	95.4%	95.17%	**95.86**%	86.89%
NaiveBayes	90.11%	**94.25**%	93.33%	86.66%
KStar	92.87%	**94.25**%	93.56%	86.66%
BFTree	95.4%	95.17%	**95.86**%	87.12%
NBTree	95.4%	95.4%	**95.86**%	86.66%
SMO(SVM)	**95.86**%	95.63%	95.63%	87.12%

As shown in Table 4, except for SMO(SVM), the highest accuracy always occurs in one of the subsets found by *ESE* for all the other classifiers. As expected, the randomly selected subset which also consists of 5 samples always has the lowest accuracy. Moreover, the decision trees built by NBTree on Subsets 1 and 2 are much simpler than that of the full sample set because of the smaller number of samples. The trees are omitted here for space restriction.

[5] http://www.cs.waikato.ac.nz/ml/weka/

Jester Data

We discussed in Section 1 that subsets of samples can also be helpful in designing customer review study. By applying our *ESE* algorithm on the Jester data, we get many sample (joke) subsets that are small and cover most markers (reviewers). Given the minimum coverage ρ, the number of qualified sample subsets and their sizes are listed in Table 5. The sample (jokes) subsets in Table 5 suggest that reviewers' ratings on a small number of objects are sufficient to retain most diversity.

Table 5. Jester Data: Number of qualified sample subsets and their sizes for given ρ

ρ	size	number of qualified subsets
0.9	5	122
0.95	8	73
0.97	10	34

According to the experiment results we presented in this section, we demonstrated that our algorithms are both efficient and effective.

6 Conclusion and Discussion

In this paper, we introduced the Parameterized Diversity Cover problem: given a sample-marker dataset and a minimum coverage threshold ρ, find the minimum sample subset that achieves coverage ρ. We propose an efficient exhaustive subset enumeration algorithm (*ESE*) which can find the optimal solution. The algorithm has two stages: (1) a greedy approach, *PGDS*, is used to first find an approximate solution for minimum subset with coverage no less than ρ; (2) an enumeration algorithm, $K\rho DS$, then searches for the optimal solution in the enumeration tree using several pruning strategies. We have evaluated the performance on three real datasets.

In the diversity cover problem, each marker is given equal weight. The problem can be extended to allow a weight to be associated with each marker. The weight can be assigned to reflect the importance of each marker and may be dynamically adjusted. For instance, groups of markers may be highly correlated. The weight of each uncovered marker is 1 before any sample is selected, and is assigned to the lowest dissimilarity of this marker to any covered marker[6]. The goal of this weighted diversity cover problem is to select samples such that the total weight of all markers is maximized.

In our future work, we will continue to investigate and evaluate alternative approaches that may offer further performance gains. An alternative greedy strategy for *PGDS* is to start from the full set of samples and remove the sample that minimizes the decrease in coverage in each subsequent step until no sample can be further removed without violating the minimal coverage requirement. Note that a similar strategy can also be employed in the $K\rho DS$ algorithm which enumerates the sample subsets that can be removed without losing more than $1 - \rho$ coverage. In some cases where a minimum subset of coverage ρ contains more than half of the samples, these alternative strategies can have a better runtime performance because they imply a smaller search space.

[6] A marker is weighted 0 if it has perfect correlation with a covered marker and is weighted 1 if it is completely independent of any covered marker.

References

[1] D. Chakrabarti, S. Papadimitriou, D. Modha, and C. Faloutsos. Fully automatic cross-associations. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2004.

[2] G. Churchill, D. Airey, H. Allayee, and *et al*. Complex trait consortium. the collaborative cross, a community resource for the genetic analysis of complex traits. *NATURE GENETICS*, 36(11):1133–7, 2004.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. Introduction to algorithms, second edition. *MIT Press and McGraw-Hill*, 2001.

[4] M. Dash and H. Liu. Feature selection for classification. *Intelligent Data Analysis*, 1(3), 1997.

[5] I. S. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic co-clustering. In *SIGKDD '03 Conference Proceedings*, 2003.

[6] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(3):133–151, 2001.

[7] K. Hartmann and M. Steel. Maximimizing phylogenetic diversity in biodiverstity conservation: greedy solutions to the noah's ark problem. *Systematic Biology*, 55(4):644–651, 2006.

[8] D. S. Hochbaum and A. Pathria. Analysis of the greedy approach in problems of maximum k-coverage. *Naval Research Quarterly*, 45:615–627, 1998.

[9] F. Y. Ideraabdullah, E. de la Casa-Espern, T. A. Bell, D. A. Detwiler, T. Magnuson, C. Sapienza, and F. P.-M. de Villena. Genetic and haplotype diversity among wild derived mouse inbred strains. *Genome Res*, 14:1880–1887, 2004.

[10] C. Jin, H. Lan, A. D. Attie, G. A. Churchill, D. Bulutuglo, and B. S. Yandell. Selective phenotyping for increased efficiency in genetic mapping studies. *Genetics*, 168(4):2285–93, 2004.

[11] F. Pan, W. Wang, A. K. Tung, and J. Yang. Finding representative sets from massive data. *IEEE International Conference on Data Mining*, 2005.

[12] A. Roberts, L. McMillan, W. Wang, J. Parker, I. Rusyn, and D. Threadgill. Inferring missing genotypes in large snp panels using fast nearest-neighbor searches over sliding windows. *ISMB*, 2007.

[13] M. Steel. Phylogenetic diversity and the greedy algorithm. *Syst. Biol*, 54:527–529, 2005.

[14] D. W. Threadgill, K. W. Hunter, and R. W. Williams. Genetic dissection of complex and quantitative traits: from fantasy to reality via a community effort. *Mamm Genome*, 13:175–178, 2002.

[15] R. W. Williams, J. Gu, S. Qi, and L. Lu. The genetic structure of recombinant inbred mice: high-resolution consensus maps for complex trait analysis. *Genome Biol*, 2(11):RESEARCH0046, 2001.

[16] Z. Xu, F. Zou, and T. J. Vision. Improving quantitative trait loci mapping resolution in experimental crosses by the use of genotypically selected samples. *Genetics*, 170(1):401–8, 2005.