SciSD: Novel Subgroup Discovery Over Scientific Datasets Using Bitmap Indices

Ohio State CSE Technical Report #OSU-CISRC-3/15-TR03, March 2015

Yi Wang Yu Su Gagan Agrawal
Computer Science and Engineering
The Ohio State University, Columbus, OH 43210
{wayi,su1,agrawal}@cse.ohio-state.edu

Tantan Liu
Google, Inc.
liutan@google.com

ABSTRACT

Subgroup discovery is a broadly applicable exploratory technique, which identifies interesting subgroups with respect to a property of interest. While there is clearly a need to apply this method to discover interesting patterns from scientific datasets comprising large-scale arrays, the existing algorithms primarily apply to relational datasets. In this paper, we present a novel algorithm, SciSD, for exhaustive but efficient subgroup discovery over array-based scientific datasets, in which all attributes are numeric. Our algorithm handles a key challenge associated with array data, which is that a subgroup identified over array data can be described based on value-based and/or dimension-based attributes. To reduce the computational costs, our SciSD algorithm extensively uses bitmap indices (and fast bitwise operations on them). We demonstrate both high efficiency and effectiveness of our algorithm by using multiple real-life datasets.

Keywords

Subgroup Discovery, Scientific Data Management, Bitmap Indexing

1. INTRODUCTION

Subgroup discovery [31, 72] is a broadly applicable exploratory technique, which identifies interesting subgroups with respect to a property of interest. The underlying goal is to extract relations with interesting characteristics between a *dependent* or *target* variable and multiple *independent* or *explaining* variables. For example, professional basketball players are mostly taller than the general population, where the attribute 'occupation' (basketball player) is the explaining variable and the attribute 'height' (greater than the average of the general population) is the target variable.

In both data management and data analytics areas, there is an increasing emphasis on (mainly) array-based scientific data. Large-scale scientific data of different sizes and dimensions are used for storing images, sensor data, simulation outputs, and statistical data in a variety of domains, including earth sciences, space sciences, life sciences, finance, and social sciences [4, 18, 39, 41, 42, 44, 49–52, 56–61, 64–71, 79, 81, 83, 84]. The existing work on subgroup discovery has been primarily restricted to application on relational or tabular datasets. In this data-driven era, it is highly desirable to customize the subgroup discovery technique to discover interesting patterns when data is stored as arrays. For example, given a set of arrays that are output from a scientific simulation, scientists may be interested in identifying the underlying relationships between variables. As a specific instance, in the output from an ocean simulation, the relationship between large value of `salinity` and other variables such as `temperature` and `depth`, is of great interest to scientists [16]. Analysis of array-based data imposes unique challenges – e.g., temperature is a *value-based* attribute that is stored as a separate array, whereas depth is a *dimension-based* attribute that corresponds to one of the array dimensions. In practice, it can be observed that the subsets of output grid points, which have a low depth and/or a high temperature are very likely to have a salinity value significantly higher than the average of the entire dataset. Clearly, the data subsets (i.e., subgroups) described by depth and/or temperature ranges can be a set of interesting subgroups, and identifying such subgroups can be of great interest. Recent work from visualization community has focused on calculating metrics such as mutual information across variables [12, 80]. However, there is no work that can identify interesting subgroups over array data automatically. A seemingly viable approach can be to apply existing *relation-based* subgroup discovery algorithms directly over array data by treating dimension-based attributes as additional relational attributes. However, this approach requires data reorganization, which can often be prohibitively expensive for large datasets, in terms of both computation and storage costs.

Developing subgroup discovery algorithms for array-based scientific data involves at least three sets of challenges. First, as we have already stated above, unlike the conventional subgroup discovered over relational data, a subgroup identified over array data can be described with value-based attributes and/or dimension-based attributes. Second, in scientific datasets, the attributes are mostly numeric, whereas most conventional subgroup discovery algorithms [7, 30, 31, 34, 37, 72] mainly target binary or categorical attributes. Third, the conventional algorithms for subgroup discovery are only able to operate with small datasets. In comparison, datasets of interest to us, particularly, from simulation datasets, can be extremely large, and are likely to become larger in the future.

In this paper, we present a novel algorithm, SciSD, for exhaustive but efficient subgroup discovery over array-based scientific datasets, in which all attributes are numeric. Our algorithm is able to

effectively discover interesting subgroups of both great generality and high quality. Moreover, our algorithm directly operates on the compact bitmap indices instead of the raw datasets, and utilizes fast bitwise operations on them, allowing processing of larger datasets.

We have demonstrated both high efficiency and effectiveness of our algorithm by using multiple real-life datasets. We first experimented on a small dataset to compare the performance (execution efficiency) as well as the quality of the output subgroups from our algorithm against SD-Map* [6], a popular subgroup discovery algorithm. The results have shown that our algorithm not only produces subgroups of significantly greater generality and higher quality, but the execution times are also lower by up to two orders of magnitude. We also evaluated our algorithms over larger datasets. We find that by using only a small number of bins, we obtain both high-quality subgroups and low execution times, which demonstrate the practicality of our algorithm.

2. SUBGROUP DISCOVERY AND ARRAY DATA

We first introduce the formal definition of subgroup discovery. Next, we introduce an example involving arrays, and extend the definition of subgroups to apply to array data. Afterwards, to process numeric attributes in array data, we discuss the quality function we choose and subgroup combination.

2.1 Preliminaries

A comprehensive overview of subgroup discovery can be found in [28]. Subgroup discovery is a combination of predictive and descriptive induction, which focuses on the extraction of relations, with respect to the property of interest given by a target variable. Unlike classification techniques for addressing classification and prediction issues, subgroup discovery describes the local patterns of data in the form of individual rules. As also formulated by Gamberger *et al.* [22] and Lavrač *et al.* [37], a rule (R) that consists of an induced subgroup description can be formally defined as:

$$R : Cond \rightarrow Target_{val} \quad (1)$$

where $Target_{val}$ is a value with respect to the property of interest given by a target variable, and $Cond$ is commonly a conjunction of attribute-value pairs given by several explaining variables. A subgroup discovery task is mainly comprised of four elements: *subgroup description language*, *quality function*, *target variable*, and *search strategy*, described in the following paragraphs. In our description, Ω_A denotes the set of all attributes, and for each attribute $a \in \Omega_A$, a value domain $dom(a)$ is defined.

DESCRIPTION 1 (SUBGROUP DESCRIPTION). *A subgroup description is defined as a conjunction of attribute-value pairs, where for each attribute-value pair $a_i = v_i$, $a_i \in \Omega_A$ and $v_i \in dom(a_i)$.*

An important measure associated with a subgroup is the *quality*, which measures the "interestingness" of the subgroup, and is used for extracting and ranking the rules. There is no consensus on the best quality function to be used for evaluating the interestingness of a subgroup.

DESCRIPTION 2 (QUALITY FUNCTION). *Let Ω_{sd} denote the universal set of all the possible subgroup descriptions, and given a particular target variable $t \in \Omega_A$, a quality function q is a mapping $q : \Omega_{sd} \times dom(t) \rightarrow R$, where R is the space of real numbers denoting quality score.*

The types of attributes including both explaining variables and target variables can be *binary*, *categorical*, or *numeric*. For processing numeric attributes, discretization is often utilized to divide a continuous value range into multiple discrete intervals.

2.2 An Example Involving Arrays

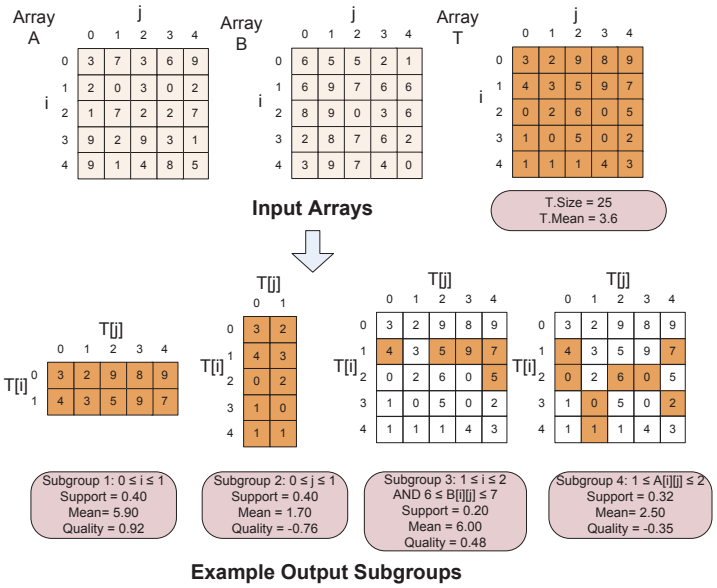


Figure 1: A Running Example of Subgroup Discovery

Our goal focuses on subgroup discovery over array-based datasets. We use an example to illustrate the idea, which is shown through Figure 1. This example involves three 2-dimensional arrays A, B, and T, where T indicates the target variable, and A and B are the two explaining variables. Two dimensions are indexed from 0 to 4, and denoted as i and j, respectively. One can view this dataset as a relational table forming 25 records, where each record has three attributes.

Our goal is to identify interesting subsets (or subgroups) of T such that each subset is significantly different from the entire array T, which has a size of 25 and a mean value of 3.6. Each subgroup is evaluated by a quality score that is based on a combination of two metrics: subgroup *size* (or *support*) and the difference with respect to the mean between the subgroup and the entire array. One can identify several subgroups of interest from the target array T.

The identified subgroups can be described by: 1) dimensional ranges of i and/or j, 2) value ranges of A and/or B, and 3) both value ranges and dimensional ranges. The $Subgroup\ 1$ and $Subgroup\ 2$, which belong to the first type of subgroups, simply select the first two rows and the first two columns of the array T, respectively. As an example of the second type of subgroup, the $Subgroup\ 4$ comprises the elements of T where the corresponding elements of A have a value between 1 and 2. The $Subgroup\ 3$ exemplifies the third type of subgroup, by selecting the elements of T within the second row and the third row, where the corresponding elements of B have a value between 6 and 7. In each of the subgroups of

interest, the mean of the value of T within the subgroup is either significantly higher or lower than the average of the entire array, and is associated with either a positive or negative quality score.

2.3 Extended Subgroup Description

Compared with the relational data model that conventional subgroup discovery algorithms operate on, one key difference of array-based scientific datasets is that, every array element is explicitly indexed by an array subscript or coordinate value in each dimension. In practice, an interesting subgroup (i.e., data subset) may also correspond to a particular spatial subarea. For example, within a 3-dimensional space modeled by `longitude`, `latitude` and `depth`, given by a target variable `salinity`, perhaps it is not only influenced by certain value-based attributes like `pressure` and `temperature`, which correspond to different arrays in the same dimensional layout, but is also correlated with the dimension-based attributes like `latitude` and `depth` that indicate array dimensions.

Formally, we categorize all the attributes possibly involved in the subgroup description into two types: *dimension-based* and *value-based*. Let Ω_A^D and Ω_A^V denote the set of all dimension-based and value-based attributes, respectively. Each dimension-based attribute a^d indicates an array dimension, and a corresponding dimensional domain $dom(a^d)$ is defined (based on dimension indices or coordinate values). Each value-based attribute a^v indicates an array name, and a corresponding value domain $dom(a^v)$ is defined based on the element values in the array. For a dimension-based attribute a_i^d, from $dom(a_i^d)$, we can obtain a set of all the possible dimensional intervals R_i^d - each interval r_i^d comprises a set of contiguous values from $dom(a_i^d)$. Similarly, for a value-based attribute a_i^v, from $dom(a_i^v)$, we can obtain a set of all the possible value intervals R_i^v, where each interval r_i^v comprises a set of contiguous values from $dom(a_i^v)$. Let R^d and R^v be the universal set of all the possible dimensional intervals and value intervals, respectively.

DESCRIPTION 3 (EXTENDED SUBGROUP DESCRIPTION). *A subgroup description is defined by a conjunction of attribute-range (or attribute-interval) pairs, where each attribute can be either dimension-based or value-based. Each dimension-based attribute-range pair is of the form $a_i^d = r_i^d$, where $a_i^d \in \Omega_A^D$, and $r_i^d \in R_i^d$, and each value-based attribute-range pair is of the form $a_i^v = r_i^v$, where $a_i^v \in \Omega_A^V$, and $r_i^v \in R_i^v$.*

2.4 Quality Function and Subgroup Combination

The initial subgroup discovery algorithms [31, 72] were proposed to facilitate the data exploration in the medical domain, and most existing subgroup discovery algorithms [19, 27, 30, 37, 82] mainly consider binary and categorical attributes. Some of these algorithms process numeric attributes by a 'straight-forward' discretization, i.e., creating a few intervals, and then applying quality function and search strategy designed for categorical attributes. Our initial attempts in applying existing algorithms for scientific simulation datasets show that such strategies are not adequate. Thus, to process numeric attributes, two major steps besides discretization are taken.

Quality Function: Most quality functions used for subgroup discovery, such as interest, novelty, significance, specificity, and Weighted Relative Accuracy (WRAcc) [28], are only applicable to binary or categorical attributes. By contrast, the quality function applicable to numeric attributes usually involves *weighing* and *aggregations* over a subgroup. In our work, we consider the *mean* with respect to the target variable as the property of interest, although other statistics can also be considered. As also used in [6], the quality function we use is referred to as *Continuous Weighted Relative Accuracy (CWRAcc)*:

$$q_{CWRAcc} = \frac{n}{N} \times (m - M) \tag{2}$$

where n and N denote the subgroup size and general population size, respectively, and m and M denote the mean with respect to the target variable in the subgroup and the general population, respectively. An alternative quality function is *Continuous Piatetsky-Shapiro (CPS)* [6]:

$$q_{CPS} = n \times (m - M) \tag{3}$$

Note that the value of the quality computed by either of these two functions can be either positive or negative, indicating m is either greater than or less than M.

Subgroup Combination: Unlike the cases where one is processing only binary or categorical attributes, it is often reasonable to combine two subgroups concerning the same attribute but with adjacent value ranges, based on a set of combination rules. Otherwise, it is very likely that we will have a massive number of subgroups, of which the majority can be further merged into the ones of greater generality. For example, two adjacent ranges $11 \leq depth \leq 30$ and $31 \leq depth \leq 50$ may be combined as a larger range $11 \leq depth \leq 50$. The combination rules we use will be discussed in Section 4.2 in details.

3. SCISD ALGORITHM

This section describes the design of our SciSD algorithm. Our algorithm extensively uses bitmap indices as a summary and efficient representation of data, instead of the actual array data. However, to simplify the presentation in this section, the algorithm will be presented as if it is operating on the array data. Use of bitmap indices to accelerate the algorithm will be described in Section 5.

3.1 Tight Optimistic Estimates

Before discussing the search strategy used in the algorithm, we first introduce the notion of *optimistic estimates*. Optimistic estimates [27] aim to safely prune the search space, especially when monotonicity with respect to the property of interest cannot be applied. An optimistic estimate of a given subgroup computes the upper bound of the quality, i.e., *the maximal quality* that can never be exceeded by any subset of this subgroup. A *tight* optimistic estimate means that, there exists a subset of the given subgroup such that the quality of this subset can reach the estimated value. Note that such a subset with the maximal quality does not necessarily correspond a subgroup description [27].

Given a subgroup sg, we denote the quality of the subgroup as $q(sg)$. Further, we denote by sg_{pos} the subset comprising all elements whose values are greater than the mean of sg. Similarly, we denote by sg_{neg} the subset comprising all elements whose values are lower than the mean of sg.

THEOREM 1. *For the Continuous Weighted Relative Accuracy quality function and a given subgroup sg, the tight optimistic esti-*

mate $oe(sg)$ can be formulated as:

$$oe(sg) = \begin{cases} q(sg_{pos}), \text{ if } q(sg) > 0 \\ q(sg_{neg}), \text{ if } q(sg) < 0 \end{cases} \quad (4)$$

Theorem 1 has been proved by Atzmüller *et al.* [6]. Our search strategy will leverage optimistic estimates to safely prune the search space, as described in the next subsection.

3.2 Search Strategy

Our SciSD algorithm performs an exhaustive search, i.e., the space of all the possible subgroups needs to be explored. Clearly, the search space is exponential in the number of attribute-range pairs. Therefore, we use certain pruning measures as well as *binning* of both dimensional and value ranges to reduce the computation costs. The search for subgroups is based on *set-enumeration tree* [10], which ensures that every node is either visited only once or not visited at all (if is pruned). The equivalence between set-enumeration trees and canonical orderings has been proved in the literature [47]. Thus, a simple rule for creating the tree is to generate the children nodes by appending only those attributes that follow all existing attributes in a given ordering. Both dimension-based and value-based attributes can be equally treated in such an ordering.

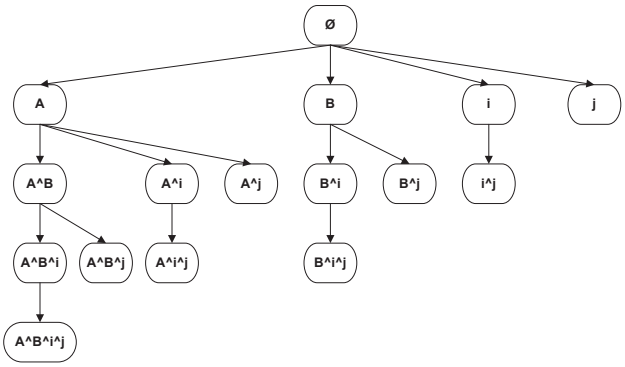


Figure 2: An Example of Set-Enumeration Tree

Figure 2 shows an example of set-enumeration tree corresponding to Figure 1 in Section 2. To keep the illustration simple, subgroup combination is not considered here. Moreover, let us assume the dimensional space of input arrays is only 2×2. Thus, both attributes i and j have two possible distinct values 0 and 1. In this tree, each node indicates not one, but a set of subgroup candidates. The set associated with each node comprises elements with conjunction over all distinct ranges of the attributes the node is associated with. For example, the node $i \wedge j$ indicates the subgroup candidates $i = 0 \wedge j = 0$, $i = 0 \wedge j = 1$, $i = 1 \wedge j = 0$, and $i = 1 \wedge j = 1$, and candidates generated by any combination of these. Ordering is maintained while generating child nodes in the tree, e.g., the node B can have children nodes $B \wedge i$ and $B \wedge j$, but not $B \wedge A$. For our explanation, we use the terms *parent subgroup* and *child subgroup* – for illustration, $A = 1$ is the parent subgroup of $A = 1 \wedge B = 2$, which is the child subgroup here.

Algorithm 1 shows our search strategy. We use two queues – *subgroup candidate queue* Q and the *output subgroup queue* Q' (lines 1 and 2). Q stores the subgroups that can potentially be 'interesting' but require further validation, and Q' stores 'interesting' subgroups after validation, i.e., the ones that are going to be a part of the output. Our search traverses the set-enumeration tree *level by level*, starting from the root node that represents the general population without any subsetting (line 3). To identify attribute-range pairs that can contribute to 'interesting' subgroups, lines 6 and 7 iterate over each attribute in a given ordering (e.g., canonical ordering), and each range in order, respectively. For each subgroup candidate in subgroup candidate queue Q, it can produce a set of child subgroups by appending another attribute-value pair, where the appended attribute follows the last attribute in its parent subgroup (line 9).

Algorithm 1: SciSD(dim-based attr. universal set Ω_A^D, val-based attr. universal set Ω_A^V, dim-range universal set R^D, val-range universal set R^V)

```
1: Let Q be a queue for searching subgroup candidates
2: Let Q' be a queue for output subgroups
3: Push the root node (represents the general population) into Q {* This
   node will be pruned later *}
4: while Q is not empty do
5:    Let sg_f be the first subgroup in Q
6:    for each attribute a_i ∈ Ω_A^D or Ω_A^V, where a_i follows the last
      attribute in the description of sg_f in the given ordering do
7:       for each range r_j ∈ R_i^D or R_i^V do
8:          Let sg_l be the last subgroup in Q
            {* Produce child subgroup candidates *}
9:          Let sg_k be the child subgroup candidate of sg_f, by adding a
            new attribute-range pair a_i = r_j
10:         if sg_l can be combined with sg_k then
11:            Update sg_l by combining it with sg_k
12:            Let sg'_l be the last subgroup in Q'
13:            sg'_l ← sg_l
14:         else if oe(sg_k) is not pruned then
15:            Push sg_k into Q
16:            if sg_k is not pruned then
17:               Push sg_k into Q'
18:            end if
19:         end if
20:      end for
21:   end for
22: end while
23: return Q'
```

For each produced child subgroup, we first consider *subgroup combination* (lines 10 to 13). Line 10 attempts to combine adjacent subgroup candidates at the same level according to certain combination rules, which will be discussed in Section 4.2. Such subgroup combination is only considered between the last subgroup candidate in the subgroup candidate queue Q and the most recently produced child subgroup.

Note that to prevent an *over-pruning* situation, i.e., a situation where two subgroups should not be pruned but their combined subgroup is pruned, we need to ensure that once two subgroups are combined, the combined subgroup will not be pruned. Lines 11 to 13 update the last subgroup in both queues Q and Q' after combination.

Subgroup pruning occurs if the tentative subgroup combination fails (lines 14 to 19). Similar to subgroup combination, a *pruning test* is applied based on certain pruning measures, which will be discussed in Section 4.1. The pruning process includes two steps. First, we check if the optimistic estimate of the subgroup can pass the pruning test (line 14). If so, the subgroup is viewed as a subgroup candidate, which is added to the subgroup candidate queue Q. By adding it to the queue Q, we can check if any of its child

subgroups can be a qualified subgroup as the search proceeds (line 15). Otherwise, this subgroup can be safely pruned because of the property of the optimistic estimate, and its child subgroups are not evaluated any further. If a subgroup qualifies because of the value of the optimistic estimate, we apply the same pruning test to the subgroup itself (line 16). Only if the subgroup still qualifies, it is added to the output subgroup queue Q' as an output subgroup (line 17). In practice, all the output subgroups can be ranked according to the quality function.

4. SUBGROUP PRUNING AND COMBINATION

In our algorithm, efficiency is maintained by subgroup pruning based on certain pruning measures, as well as subgroup combination rules based on a set of combination rules, which are explained in this section.

4.1 Pruning Measures

Our algorithm uses three pruning measures to determine if a subgroup should be pruned or not. Two of them are relatively straightforward - a minimum support (or size) of the subgroup and a minimum threshold quality, since a subgroup of interest should be large enough and significantly different from the general population.

PRUNING MEASURE 1 (MINIMUM SUPPORT). *Given a subgroup sg, with $sup(sg) = \frac{n}{N}$, where n and N are the size of sg and the general population, respectively, sg is pruned if*

$$sup(sg) < sup_{min}$$

PRUNING MEASURE 2 (MINIMUM ABSOLUTE QUALITY). *Given a subgroup sg, sg is pruned if*

$$|q(sg)| < q_{min}$$

The third pruning measure we use is as follows – compared with the parent subgroup, a child subgroup should be considered only if it has a higher quality than the parent subgroup. The motivation is that if such a pruning rule is not used, a large number of small and redundant subgroups will be generated by the algorithm.

PRUNING MEASURE 3 (MINIMUM RELATIVE QUALITY). *Given a subgroup sg_{parent} and its child subgroup sg_{child}, sg_{child} is of interest only if*

$$\begin{cases} q(sg_{child}) - q(sg_{parent}) \geq q^r_{min}, & \text{if } q(sg_{parent}) > 0 \\ q(sg_{parent}) - q(sg_{child}) \geq q^r_{min}, & \text{if } q(sg_{parent}) < 0 \end{cases}$$

According to Algorithm 1, for the subgroup candidates that involve multiple attribute-range pairs, their non-root parent subgroups must have already satisfied Pruning Measure 2. By using Theorem 2, Pruning Measure 3 here can be considered as a *sufficient* condition of Pruning Measure 2 for these subgroup candidates.

THEOREM 2. *Given a parent subgroup sg_{parent} that satisfies Pruning Measure 2, and its child subgroup sg_{child}, if both sg_{parent} and sg_{child} satisfy Pruning Measure 3, then sg_{child} also satisfies Pruning Measure 2.*

PROOF. If $q(sg_{parent}) > 0$, according to Pruning Measures 2 and 3, we have $q(sg_{parent}) \geq q_{min}$, and $q(sg_{child}) - q(sg_{parent}) \geq q^r_{min}$. Thus,

$$\begin{aligned} q(sg_{child}) &\geq q(sg_{parent}) + q^r_{min} \\ &> q(sg_{parent}) > q_{min} \end{aligned}$$

If $q(sg_{parent}) < 0$, the proof proceeds analogously. □

4.2 Combination Rules

Three combination rules are defined to determine if two subgroups can be combined or not – all three conditions must be met for two subgroups to be combined. First, intuitively, only the two subgroups, which belong to the same parent subgroup and are adjacent at the same level in the set-enumeration tree, can be combined. In other words, one of the prerequisites of subgroup combinability is the *existence of adjacency*.

COMBINATION RULE 1 (EXISTENCE OF ADJACENCY). *Given two subgroups described by two conjunctions of attribute-range pairs $a_1 = r_1 \wedge a_2 = r_2 ... \wedge a_n = r_n$ and $a'_1 = r'_1 \wedge a'_2 = r'_2 ... \wedge a'_n = r'_n$, respectively, then the two subgroups are considered adjacent and can be combined only if $\forall i, 1 \leq i \leq n : a_i = a'_i$, $\forall i, 1 \leq i < n : r_i = r'_i$, and, r_n and r'_n are two adjacent dimensional ranges in R^D_n or value ranges in R^V_n.*

Second, as also mentioned earlier in Section 3.2, if any two adjacent subgroups can pass the pruning test, they may still be pruned after combination, due to the low quality of the combined subgroup. This should be avoided. Therefore, it is necessary to set a combination rule to ensure that the combined subgroup should pass the pruning test, i.e., one combination rule should be a *sufficient* condition of satisfying all the three pruning measures defined in Section 4.1. Note that as Algorithm 1 shows, one of the subgroups used for combination is from the output subgroup queue, and it has already passed the pruning test. Here, we propose an easy-to-compute criterion referred to as *difference homogeneity*, which is defined as Combination Rule 2. A positive value of difference homogeneity indicates that the means with respect to the target variable in these two subgroups are homogeneous, i.e., either greater than or less than the mean of the general population at the same time.

COMBINATION RULE 2 (DIFFERENCE HOMOGENEITY). *Let m_1 and m_2 be the mean with respect to the target variable in two given subgroups, and let M be the mean of the general population, then the two subgroups can be combined only if*

$$(m_1 - M) \times (m_2 - M) > 0$$

Formally, given two subgroups sg_1 and sg_2 which are combined into a subgroup $sg_{combined}$, and sg_1 has passed the pruning test, the following three theorems ensure that a combined subgroup should never be pruned, by satisfying all the three pruning measures defined in Section 4.1.

THEOREM 3. *If sg_1 and sg_2 are combined into $sg_{combined}$, and sg_1 has passed the pruning test, then $sup(sg_{combined}) > sup_{min}$.*

PROOF. Straight-forward. □

THEOREM 4. *If sg_1 and sg_2 are combined into $sg_{combined}$, and sg_1 has passed the pruning test, then $|q(sg_{combined})| \geq q_{min}$.*

PROOF. Let $n_{combined}$ and $m_{combined}$ be the size and mean of $sg_{combined}$, respectively. By using Equation 2, we can have

$$\begin{aligned} q(sg_{combined}) &= \frac{n_{combined}}{N} \times (m_{combined} - M) \\ &= \frac{n_1 + n_2}{N} \times (\frac{m_1 \times n_1 + m_2 \times n_2}{n_1 + n_2} - M) \\ &= \frac{(m_1 - M) \times n_1 + (m_2 - M) \times n_2}{N} \\ &= q(sg_1) + q(sg_2) \end{aligned} \quad (5)$$

For sg_1, $|q(sg_1)| \geq q_{min}$. Since Combination Rule 2 implies $q(sg_1) \times q(sg_2) > 0$, by using Equation 5, we can have

$$\begin{aligned} |q(sg_{combined})| &= |q(sg_1) + q(sg_2)| \\ &> |q(sg_1)| > q_{min} \end{aligned}$$

□

THEOREM 5. *If sg_1 and sg_2 are combined into $sg_{combined}$, and sg_1 has passed the pruning test. Let the parent subgroup of sg_1 and sg_2 be sg_{parent}, then*

$$\begin{cases} q(sg_{combined}) - q(sg_{parent}) \geq q^r_{min}, & \text{if } q(sg_{parent}) > 0 \\ q(sg_{parent}) - q(sg_{combined}) \geq q^r_{min}, & \text{if } q(sg_{parent}) < 0 \end{cases}$$

PROOF. If $q(sg_{parent}) > 0$, for sg_1,

$$q(sg_1) \geq q(sg_{parent}) + q^r_{min} > 0$$

Since Combination Rule 2 implies $q(sg_1) \times q(sg_2) > 0$, we can have $q(sg_2) > 0$. By using Equation 5,

$$\begin{aligned} q(sg_{combined}) - q(sg_{parent}) &= q(sg_1) + q(sg_2) - q(sg_{parent}) \\ &\geq q^r_{min} + q(sg_2) > q^r_{min} \end{aligned}$$

If $q(sg_{parent}) < 0$, the proof proceeds analogously. □

Lastly, in practice it is helpful for the users that a subgroup does not spread too broadly. Therefore, it is necessary to evaluate the *distribution purity* of a combined subgroup. Before introducing the next combination rule, we need to propose another criterion referred to as *Continuous Weighted Entropy* (CWE), which is adapted from the entropy used for evaluating the purity of classification:

$$CWE = -\sum_{i=1}^{n} p(i) \times \log p(i) \times \frac{|v_i - M|}{v_n - v_1} \quad (6)$$

where n is the number of intervals for the target variable, $p(i)$ and v_i denote the probability and representative value of the ith interval, respectively, and M represents the mean of the general population. Since v_1 and v_n are the minimum and maximum representative value, respectively, $\frac{v_i - M}{v_n - v_1}$ here represents the $weight$, which implies the *normalized distance* from the ith interval to the mean of the general population. Therefore, for two subgroups of equal size, the CWE of a subgroup in which most elements are within the intervals remote from the mean of the general population, tends to be greater than, the other subgroup in which most elements are within the intervals close to the mean of the general population.

The third combination rule aims to restrict the extent to which adjacent subgroups can be combined, by introducing a user-specified parameter – *maximum continuous weighted entropy CWE_{max}*.

COMBINATION RULE 3 (DISTRIBUTION PURITY). *If two given subgroups are combined into a subgroup $sg_{combined}$, and let its continuous weighted entropy be $CWE(sg_{combined})$, then the two subgroups can be combined only if*

$$CWE(sg_{comb}) \leq CWE_{max}$$

This combination rule can effectively control the *granularity* of output subgroups and hence help the user to control the number of output subgroups at a manageable level. Generally, a larger maximum CWE value tends to generate fewer subgroups yet of greater generality and higher quality.

5. ALGORITHM OPTIMIZATION USING BITMAPS

In this section, we first provide background information on bitmap indexing, and then we discuss the algorithm acceleration using bitmaps. Note that our actual algorithm implementation is based entirely on bitmaps. For clarity in our write-up, we are presenting bitmaps as an optimization mechanism.

5.1 Background: Bitmap Indexing

Bitmap indexing, which utilizes the fast bitwise operations supported by the computer hardware, has been shown to be an efficient approach for querying static (i.e., read-only or append-only) data, and has been initially used in data warehouse [46, 77, 78] and recently in scientific data management [53, 54, 73]. Particularly, recent work has shown that bitmap indexing can help support efficient querying of scientific datasets stored in native formats [17, 61]. Our subgroup discovery algorithm involves a novel use of bitmap indices. Thus, before presenting our algorithm, we provide background on bitmaps, and then describe how they are customized for our application.

ID	Value	e_0	e_1	e_2	e_3	e_4	e_5	i_0	i_1	i_2
		=1	=2	=3	=4	=5	=6	[1, 2]	[3, 4]	[5, 6]
0	5	0	0	0	0	1	0	0	0	1
1	4	0	0	0	1	0	0	0	1	0
2	2	0	1	0	0	0	0	1	0	0
3	5	0	0	0	0	1	0	0	0	1
4	6	0	0	0	0	0	1	0	0	1
5	1	1	0	0	0	0	0	1	0	0
6	3	0	0	1	0	0	0	0	1	0
7	1	1	0	0	0	0	0	1	0	0
Dataset		Low Level Indices						High Level Indices		

Figure 3: An Example of Bitmap Indexing

Figure 3 shows an example of bitmap indexing. In this example, the dataset contains a total of 8 elements with 6 distinct values. The *low-level* bitmap indices contain 6 bitvectors, where each bitvector corresponds to one value. The number of bits within each bitvector is the same as total number of elements in the dataset. In each bitvector, a bit is set to 1 if the value for the corresponding data

element value is equal to the *bitvector value*, i.e., the particular distinct value for which this vector is created. The *high-level* indices can be generated based on certain binning strategies [5], and here 3 high-level indices are built.

This simple example only contains integer values. Bitmap indexing also has been shown to be an efficient method for floating-point values [75]. For such datasets, instead of building a bitvector for each distinct value, we can first group a set of values together by certain binning strategies, and build bitvectors for these bins. This way, the total number of bitvectors can be kept at a manageable level. From this example we can also see that the number of bits within each level of bitmap indices is $n \times m$, where n is the total number of elements and m is the total number of bitvectors. This can result in sizes even greater than the size of the original dataset, causing high time and space overheads for index creation, storage, and query processing. To solve this problem, run-length compression algorithms such as Byte-aligned Bitmap Code (BBC) [5] and Word-Aligned Hybrid (WAH) [74] have been developed to reduce the bitmap size. The main idea of these approaches is that for long sequences of 0s and 1s within each bitvector, an encoding is used to count the number of continuous 0s or 1s. Such encoded counts are stored, requiring less space. Another property of the run-length compression methods is that, it supports fast bitwise operations without decompressing the data.

Bitmaps are widely used for supporting queries on scientific datasets [17, 73]. Thus, before our algorithm is used, it is quite likely that bitmaps are already available, and hence we assume no preprocessing cost for generating bitmaps in our experiments. Further, even if bitmaps are not available, with an appropriate number of bins (e.g., fewer than 1000), bitmap indexing is still cheap in terms of both computation and storage. Construction of bitmaps only took around 2 minutes per GB of original data, per core, in our experiments, since it requires at most two full scans of raw data. The size of the indices ranged between 15% and 30% of the size of the original dataset.

5.2 Algorithm Acceleration Using Bitmaps

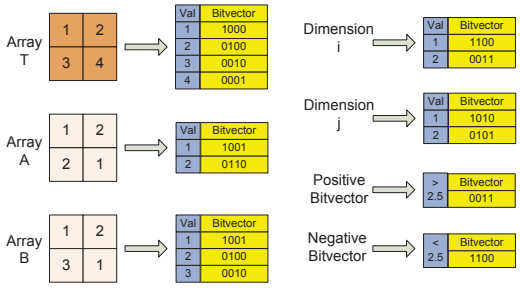


Figure 4: Use of Bitmaps in SciSD

A key characteristic of our algorithm is that, the input is compact bitmap indices instead of the original raw datasets. Note that bitmap (indices) are not used in the conventional way indices are used, i.e., for querying actual data more efficiently. Instead, bitmaps are used as a summary representation of the data. Also note that bitmaps loose a certain level of precision because of binning for value-based attributes. However, to control costs, our subgroup discovery method must use binning of value-based and dimension-based attributes.

A bitmap example is shown in Figure 4. Besides the normal representation of the data with bitmaps, our algorithm uses two other types of bitmaps. Returning to our running example, assume that within a 2×2 space, 2 dimensions (i.e., dimension-based attributes) row and column, are denoted as i and j, respectively. The subgroup discovery involves 3 arrays (i.e., value-based attributes) T, A, and B, where T is the target variable. All the bitmap indices here are generated in a row-major fashion. Three types of bitmap indices are generated in this example: 1) (conventional) indices for value-based attributes, e.g., T, A and B; 2) indices for dimension-based attributes, e.g., i and j; and 3) two *sign bitvectors* – both a *positive bitvector* and a *negative bitvector*.

These bitvectors are used in the following fashion. With the first two types of indices, each attribute-range pair can be represented by a bitvector in the bitmap, whether the attribute is dimension-based or value-based. Particularly, the second type of indices are created in the same way as value-based attributes are indexed, i.e., by treating each coordinate value as an array element value. Equi-depth binning [5], which partitions the entire domain in such a way that each bin contains an approximately the same number of elements, is used for indexing both value-based and dimension-based attributes. Use of binning restricts the ranges (of both values and dimensions) that can be used in describing a subgroup. However, this has a very significant impact on the execution time of the algorithm, as both the size of bitvectors and number of potential subgroups to be considered is now restricted to a manageable level.

The two sign bitvectors are used for optimistic estimates. Particularly, the positive bitvector indicates all the elements greater than the mean with respect to target variable in the original dataset (2.5 in this example), and leads to a positive quality. Similarly, the negative bitvector results in a negative quality.

Compared with conventional bitmap indexing that only outputs type-1 indices, our approach generates more indices. However, it turns out that both the bitvectors for dimension-based attributes and two individual sign bitvectors do not add much space or time complexity. This is because of the regularity of array elements within the same dimensional range, leading to very high compression ratios (e.g., 1%).

Now, we explain how efficiency is achieved in our algorithm using bitmaps. First, since the size of bitmap indices is often only 15%-30% of the original data size, I/O performance can be clearly improved. In addition, specific algorithm steps are accelerated as follows. First, a conjunction of multiple attribute-range pairs can be calculated by bitwise AND operations over different bitvectors. As each attribute-range pair corresponds to a bitvector in our bitmaps, a subgroup described by a conjunction of multiple attribute-range pairs can now be represented by a single bitvector after bitwise AND operations. This property is particularly useful every time a parent subgroup produces a child subgroup by appending another attribute-range pair (see line 9 in Algorithm 1). In our running example, $A = 1 \wedge i = 2$ can be derived by $1001 \wedge 0011 = 0001$.

Next, a disjunction of two attribute-range pairs can be obtained by bitwise OR operation between two bitvectors. This can speed up the subgroup combination process in our algorithm (see line 11 in Algorithm 1), since the combination of two attribute-range pairs can be viewed as a disjunction. In the above example, $B <= 2$ can be derived from $1001 \vee 0100 = 1101$.

Another benefit of bitmaps is that, both subgroup size and the mean value with respect to the target variable in a subgroup can be computed efficiently with bitmaps. This property can accelerate all the computations involved in our quality evaluations, pruning measures, and combination rules. As the membership of a subgroup can be described by a single bitvector, the subgroup size can be easily obtained by counting the number of 1s in the bitvector. Moreover, if we denote the bitmap that corresponds to the target variable as the *target bitmap*, associated with the target bitmap, the mean with respect to the target variable in a subgroup can be calculated as follows:

$$m = \frac{\sum_{i=1}^{n} v_i \times COUNT(b_i \wedge b)}{\sum_{i=1}^{n} COUNT(b_i \wedge b)} \quad (7)$$

where m is the mean with respect to the target variable in the subgroup, n is the number of bitvectors in the target bitmap, v_i is the representative value of the bitvector b_i in the target bitmap, and b is the bitvector that represents the membership of the subgroup. It turns out that the mean value can be calculated efficiently and with reasonably high accuracy based on the above process. In the above example, as the subgroup $A = 1$ can be represented by the bitvector 1001, the mean with respect to T in this subgroup can be computed by $\frac{1 \times COUNT(1000 \wedge 1001) + 4 \times COUNT(0001 \wedge 1001)}{COUNT(1000 \wedge 1001) + COUNT(0001 \wedge 1001)} = 2.5$.

Finally, the two *sign bitvectors* can facilitate the optimistic estimates discussed in Section 3.1 (as used in line 14 of Algorithm 1). In Equation 4, given a bitvector that indicates all the elements in a given subgroup sg, a simple bitwise AND operation between this bitvector and the *positive bitvector* can result in the bitvector corresponding to sg_{pos}. Similarly, sg_{neg} can be efficiently calculated with the *negative bitvector*. In the above example, to obtain the subset with element values greater than the mean in the subgroup $A = 1$ represented by the bitvector 1001, we can perform a bitwise AND between this bitvector and the positive bitvector, $1001 \wedge 0011 = 0001$.

6. EXPERIMENTAL RESULTS

In this section, we evaluate the performance (execution efficiency) of our algorithm as well as the quality of the output subgroups. We designed the experiments with the following goals: 1) to compare our algorithm with SD-Map* [6], which is a popular subgroup discovery algorithm (applicable to relational data involving numerical attributes) – it has been implemented in an open-source software VIKAMINE [1], and 2) to demonstrate both performance of our algorithm and the quality of the output subgroups, by varying both the maximum continuous weighted entropy (CWE) and the number of bins used in bitmap indexing.

6.1 Experimental Setup

Our experiments were conducted using four real-life scientific datasets, which are all stored in NetCDF, one of the popular array formats. The first two datasets are downloaded from the World Ocean Atlas 2009 (WOA09) [2] monthly compisiting data. Because these two datasets are generated on 5° and 1° grids, we refer to them as WOA09_5DEG and WOA09_1DEG, respectively. Each dataset comprises five attributes including `apparent oxygen utilization (AOU)`, `temperature`, `dissolved oxygen (DO)`, `salinity`, and `oxygen saturation (OS)`, in a 4-dimensional space, which is modeled by `longitude`, `latitude`, `depth`, and `time`. The sizes of WOA09_5DEG and WOA09_1DEG datasets are 14 MB and 373 MB, respectively. The third dataset is obtained from World Ocean Atlas 2013 (WOA13) [3] annual compisiting data, which is generated on 5° grids. Compared with the first two datasets, this dataset is modeled with the same four dimensions, but it comprises three more attributes – `silicate`, `phosphate`, and `nitrate`. We refer to this dataset as WOA13. The fourth dataset is generated by Parallel Ocean Program (POP) [29]. POP is an ocean circulation model, and the execution we used has a grid resolution of approximately 10 km (horizontally), and vertically, it has a grid spacing of nearly 10 m near the surface, and reaching 250 m in the deep ocean. The dataset mainly comprises four attributes, `salinity`, `temperature`, `UVEL` and `VVEL`, where UVEL and VVEL observe the velocity of ocean current in the grid-x and grid-y directions, respectively. POP generates 1.4 GB data for each attribute per time-slice, and each attribute is modeled with three dimensions: `latitude`, `longitude`, and `depth`. We only used one time-slice which is referred to as the `POP` dataset in our experiments, and thus, the total data size of the fourth dataset was 5.6 GB. The number of attributes involved in our experiments is up to 8, which we believe is a large number compared to the number of attributes likely of interest to any given scientist. Our experiments were conducted on a machine with 8 GB of main memory and Intel(R) Xeon(R) 2.53 GHz CPU. The version of VIKAMINE we used was 2.2.

6.2 Comparison with SD-Map*

Our first set of experiments compared the performance of our algorithm and the quality of its output subgroups against SD-Map*, a popular subgroup discovery algorithm over numeric attributes. SD-Map* has been implemented in an open-source software VIKAMINE, and it uses the same quality function – Continuous Weighted Relative Accuracy (CWRAcc), which is defined by Equation 2. Although we are also aware of a small number of other algorithms that can work with numeric attributes, we cannot directly compare with them, because they either use different quality functions, and/or their implementations were not available to us. Since VIKAMINE can only support subgroup discovery over small relational datasets, we only used WOA09_5DEG and WOA13 these two datasets, which have a size of 14 MB and 8 MB, respectively. To match the format expected by VIKAMINE, we converted the data from NetCDF to the CSV format. In the process, the four dimension-based attributes were also added as additional columns, since VIKAMINE can only process relational data. Note that as mentioned in Section 1, treating array dimensions as additional relational attributes is not a practical approach for large datasets, due to the high data reorganization and storage costs.

First, among the five value-based attributes in WOA09_5DEG dataset, we set `OS` as the target variable – its mean value is 71.03. We used 100 bins for indexing each value-based attribute, and 12 bins for indexing each dimension-based attribute. The minimum support, minimum absolute quality, and minimum relative quality were set as 0.01, 0.09, and 0.001, respectively. The maximum CWE was varied from 0.25 to 1.5.

Tables 1 and 2 show the best 8 subgroups discovered by SD-Map* and our algorithm on WOA09_5DEG dataset, respectively, with the maximum CWE of 1.5. The subgroups within each table are in descending order of absolute quality measured by CWRAcc.

We can make the following four observations about the two algorithms. First, our algorithm can discover more interesting subgroups (i.e., subgroups of higher quality). For example, the quality of the best 4 subgroups discovered by our algorithm is almost twice the best subgroup discovered by SD-Map*. This is because, with the use of Equation 5, the subgroup combination mechanism

Table 1: The Best 8 Subgroups Discovered on WOA09_5DEG by SD-Map*

Rank	Subgroup	CWRAcc	Support	Mean
1	AOU $<$ 0.08	6.24	0.20	102.24
2	depth $<$ 40	6.08	0.21	99.81
3	AOU $<$ 0.08 AND depth $<$ 40	4.65	0.15	102.45
4	DO $\geq$ 6.14	4.63	0.20	94.18
5	0.08 $\leq$ AOU $<$ 0.75	4.62	0.20	94.13
6	temperature $>$ 18.41	4.55	0.20	93.79
7	4.98 $\leq$ DO $<$ 6.14	3.85	0.20	90.29
8	temperature $>$ 18.41 AND AOU $<$ 0.08	3.56	0.11	102.14

Table 2: The Best 8 Subgroups Discovered on WOA09_5DEG by SciSD with Maximum CWE of 1.5

Rank	Subgroup	CWRAcc	Support	Mean
1	-0.43 $\leq$ AOU $\leq$ 1.95	12.26	0.58	92.17
2	2.68 $\leq$ AOU $\leq$ 7.99	-12.25	0.34	35.00
3	0.01 $\leq$ DO $\leq$ 4.29	-11.61	0.36	38.78
4	4.43 $\leq$ DO $\leq$ 10.57	11.44	0.60	90.10
5	0 $\leq$ depth $\leq$ 200	9.30	0.51	89.34
6	250 $\leq$ depth $\leq$ 1500	-9.30	0.49	52.14
7	2.36 $\leq$ temperature $\leq$ 12.97	-6.57	0.50	57.89
8	16.16 $\leq$ temperature $\leq$ 32.68	5.15	0.25	91.64

in our algorithm leads to subgroups that are larger and of higher quality. Second, SD-Map* is only able to discover the subgroups with the mean value significantly greater than the general population, i.e., the subgroups with a positive CWRAcc. Our algorithm can also discover interesting subgroups with a negative CWRAcc. This is because our Pruning Measure 2 is also able to capture interesting subgroups with significantly smaller means. Third, as Table 1 shows, a nontrivial fraction of subgroups generated by SD-Map* are redundant. $Subgroup$ 3 is really the intersection of $Subgroup$ 1 and $Subgroup$ 2, and in fact, has a lower CWRAcc than either of its two super-subgroups. Similarly, $Subgroup$ 8 is an analogous intersection of $Subgroup$ 1 and $Subgroup$ 6. In contrast, our algorithm does not result in any subgroup that is redundant. This is because our Pruning Measure 3 can prevent any subgroup specialization that does not lead to quality gain. Finally, the subgroups discovered by our algorithm tend to have greater generality, i.e., larger subgroup size, while we observed that most subgroups discovered by SD-Map* could have been merged into larger subgroups. For example, $Subgroup$ 1 and $Subgroup$ 5, as well as $Subgroup$ 4 and $Subgroup$ 7 in Table 1, could have been combined into a larger subgroup. In comparison, our algorithm can combine as many subgroups as possible, under the user-specified maximum CWE. Therefore, our algorithm can ensure that any two of the output subgroups cannot be further combined, facilitating better exploration from scientists.

Table 3 summarizes the statistics of the subgroups discovered by SD-Map* and our algorithm with varying maximum CWE on WOA09_5DEG dataset. 'SciSD_0.25', 'SciSD_0.5', 'SciSD_1', and 'SciSD_1.5' are denoted as the algorithm with the maximum CWE of 0.25, 0.5, 1, and 1.5, respectively. These statistics include the number of output subgroups, execution times, average support, average CWRAcc, and the number of attributes involved per subgroup. Note that the reported execution times of SD-Map* does not include the data preprocessing time, i.e., the data conversion time.

Table 3: Comparison between SD-Map* and SciSD on WOA09_5DEG with Varying Maximum CWE

Algorithm	# of Subgroups	Exe Times (secs)	CWRAcc	Support	# of Attr.
SciSD_0.25	111	58.61	0.75	0.04	1.05
SciSD_0.5	88	48.53	1.03	0.06	1.06
SciSD_1	53	32.67	1.90	0.12	1.10
SciSD_1.5	26	17.37	4.17	0.27	1.19
SD-Map*	1792	2116	0.48	0.02	3.06

We can make three observations. First, we can see that, our algorithm not only produces subgroups of greater generality and higher quality than SD-Map*, but the execution times are also lower by a factor of up to two orders of magnitude. Second, our algorithm manages to restrict the number of subgroups within a manageable level, whereas SD-Map* reports a large number of subgroups, which tend to be too specialized. A prominent example is that, with respect to the attribute `AOU`, our algorithm with the maximum CWE of 1.5 only discovered two subgroups, with positive and negative quality, respectively. In comparison, SD-Map* discovered 766 subgroups, and all of them turn out to be overlapping subsets of the only subgroup with the positive quality discovered by our algorithm (this subgroup is the $Subgroup$ 1 in Table 2). Note that VIKAMINE can support a post-processing module named 'minimal improvement filter', which aims to filter out redundant subgroups and minimize the final output, but such post-processing currently is not supported over numeric attributes. Lastly, we can see that maximum CWE can be used to effectively control the granularity of the output subgroups. Generally, a larger maximum CWE can allow more subgroups to be combined, leading to fewer output subgroups, which are of greater generality and higher quality. A larger maximum CWE results in lower execution times also.

Next, we experimented on another dataset WOA13 by setting `nitrate` as the target variable – its mean value is 6.32. We used 100 bins for indexing each value-based attribute, and 12 bins for indexing each dimension-based attribute, with an exception of `time` dimension. Only 1 bin was used for indexing `time`, because this annual compositing dataset only comprises a single time slice. Our method set the minimum support, minimum absolute quality, and minimum relative quality as 0.002, 0.14, and 0.001, respectively. The maximum CWE was varied from 0.25 to 1.5.

Table 4: The Best 8 Subgroups Discovered on WOA13 by SD-Map*

Rank	Subgroup	CWRAcc	Support	Mean
1	1.16 $\leq$ phosphate	3.22	0.20	22.40
2	1.16 $\leq$ phosphate AND time $=$ 1	3.22	0.20	22.40
3	12.75 $\leq$ silicate AND 1.16 $\leq$ phosphate	2.66	0.15	23.74
4	12.75 $\leq$ silicate AND 1.16 $\leq$ phosphate AND time $=$ 1	2.66	0.15	23.74
5	12.75 $\leq$ silicate	2.65	0.20	19.61
6	12.75 $\leq$ silicate AND time $=$ 1	2.65	0.20	19.61
7	1.16 $\leq$ phosphate AND 15.50 $\leq$ depth	0.84	0.05	23.30
8	1.16 $\leq$ phosphate AND 15.50 $\leq$ depth AND time $=$ 1	0.84	0.05	23.30

Tables 4 and 5 show the best 8 subgroups discovered by SD-Map* and our algorithm (with the maximum CWE of 1.5) on WOA13 dataset, respectively. We can still draw the same conclusions as those from the previous experiment. Particularly, in Table 4, we can see that the best 8 subgroups discovered by SD-Map* are actu-

Table 5: The Best 8 Subgroups Discovered on WOA13 by SciSD with Maximum CWE of 1.5

Rank	Subgroup	CWRAcc	Support	Mean
1	phosphate $<$ 0.84	-3.41	0.74	1.71
2	0.88 $\leq$ phosphate $<$ 3.39	3.40	0.25	19.90
3	silicate $<$ 8.45	-2.66	0.75	2.76
4	13.90 $\leq$ silicate $<$ 185.23	2.65	0.19	20.20
5	40 $\leq$ depth $<$ 175	-0.98	0.25	2.35
6	400 $\leq$ depth $<$ 1050	-0.81	0.20	2.24
7	0 $\leq$ depth $<$ 35	0.80	0.08	16.17
8	200 $\leq$ depth $<$ 375	0.76	0.06	19.39

Table 6: Comparison between SD-Map* and SciSD on WOA13 with Varying Maximum CWE

Algorithm	# of Subgroups	Exe Times (secs)	CWRAcc	Support	# of Attr.
SciSD_0.25	37	17.69	0.39	0.059	1
SciSD_0.5	33	14.64	0.69	0.180	1
SciSD_1	19	10.27	1.00	0.189	1.16
SciSD_1.5	16	6.12	1.06	0.140	1.30
SD-Map*	3036	508	0.24	0.017	4.28

ally all covered by either $Subgroup$ 1 or $Subgroup$ 5. The other 6 subgroups are all subsets of these two subgroups. In comparison, among the best 8 subgroups discovery by our algorithm, if we only consider the subgroups with a positive CWRAcc, $Subgroup$ 2 in Table 5 has higher quality than $Subgroup$ 1 in Table 4, and $Subgroup$ 4 in Table 5 has no lower quality than $Subgroup$ 5 in Table 4.

Similar to Table 3, Table 6 summarizes the comparison results on WOA13 dataset. It turns out that we can make the similar observations as those from the previous experiment. First, the average quality of the output subgroups discovered by our algorithm is 1.6x - 4.4x of SD-Map*'s output. Second, our execution time is only 1.2% of SD-Map*, when a large maximum CWE (1.5) is used. This is because lenient subgroup combination can effectively reduce the processing costs. Third, compared with SD-Map*, which produces a large number of redundant and meaningless subgroups, our output subgroups are much more compact. For instance, among the total 3036 output subgroups, the attribute `phosphate` occurs 2187 times, but only in the attribute-range pair '1.16 $\leq$ phosphate', which corresponds to $Subgroup$ 1 in Table 4. This implies that the other 2186 subgroups (or 72% of output) involving `phosphate` are all redundant. In addition, 50% of the output subgroups discovered by SD-Map* are simply created by involving one more attribute-range pair $time = 1$, which does not help improve the quality.

6.3 Evaluation over Larger Datasets

We next experimented on the two larger datasets – WOA09_1DEG and POP datasets. In this set of experiments we varied the maximum CWE and the number of bins used for indexing value-based attributes. One of our main goals was to understand how binning (a critical step in using bitvectors) impacts the execution time and the output quality of the algorithm. We set the minimum support, minimum absolute quality, and minimum relative quality as 0.01, 0.04, and 0.001, respectively. For WOA09_1DEG dataset, the maximum CWE was varied to be 0.25, 0.5, 1, and 1.5. The number of bins used for indexing each of the value-based attributes was varied between 20, 40, 60, 80, and 100. The number of bins used for indexing dimension-based attributes – `latitude`, `longitude`, `depth`, and `time`, was fixed throughout all the experiments, and was 18, 36, 24, and 12, respectively, for these four dimensions. For POP dataset, the maximum CWE was varied from 0.25 to 1. The number of bins used for indexing each of the value-based attributes was varied from 20 to 60. The number of bins used for indexing the dimension-based attributes `latitude`, `longitude`, and `depth` was also fixed – 24, 36, and 21, respectively, for these three dimensions.

Figures 5 show the evaluation results on WOA09_1DEG and POP datasets. First, we can see that using a larger maximum CWE and fewer bins can result in fewer subgroups being output. This is because a larger maximum CWE allows more small subgroups to be combined, and fewer bins lead to more coarse-grained binning and fewer attribute-range pairs. Second, in most cases, a larger maximum CWE and use of fewer bins also lead to lower processing times. The reason is two-fold. On one hand, more lenient subgroup combination can help eliminate more search space where multiple attributes are involved. On the other hand, use of fewer bins also implies fewer attribute-range pairs involved in the subgroup discovery. Finally, for both datasets, we can see higher average quality with larger maximum CWE and smaller number of bins. However, note that the subgroup quality cannot always be improved by increasing maximum CWE or decreasing the number of bins, since such quality is also highly dependent on the dataset itself. Two exceptions can be found in Figure 5(c). The first exception is that, when the number of bins is 20, using the maximum CWE of 1.5 cannot result in a subgroup of quality higher than using the maximum CWE of 1. The other exception is that, when the number of bins is greater than 20 and the maximum CWE is 1.5, the subgroup quality cannot be further improved with more bins. However, overall, high-quality results and reduced execution times with 20 bins point to the practicality of our algorithm, i.e., it can be effective on large datasets and can provide reasonable response times.

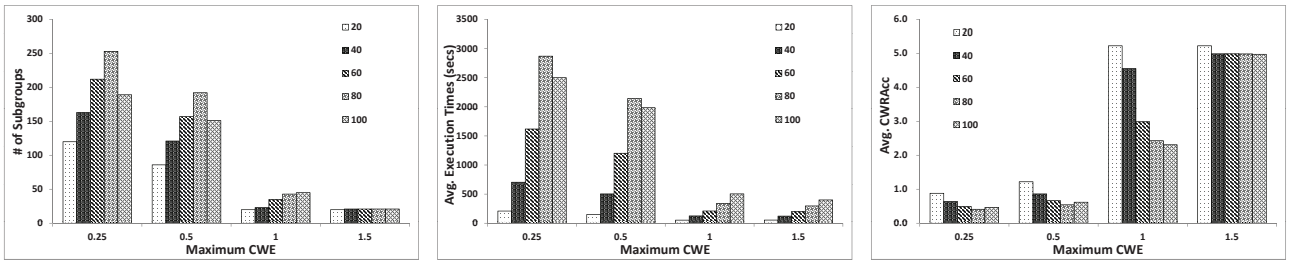
6.4 Effectiveness of Search Strategy

Table 7: Search Efficiency Evaluation on WOA09_1DEG with Varying # of Bins and Maximum CWE of 1.5

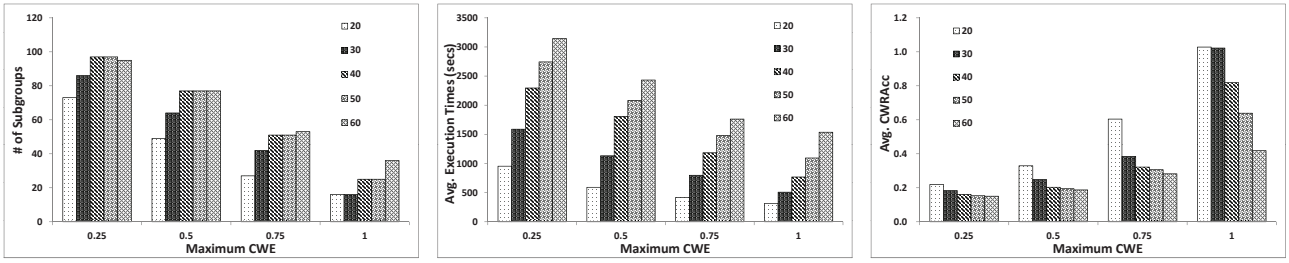
# of Bins	20	40	60	80	100
Size of Search Space	3.0E+10	4.8E+11	2.4E+12	7.6E+12	1.9E+13
# of Visited Subgroups	1626	2216	2676	3136	3596
# of Pruned Subgroups	1470 (90.4%)	1984 (89.5%)	2366 (88.4%)	2752 (87.8%)	3139 (87.3%)
# of Combined Subgroups	136 (8.4%)	211 (9.5%)	289 (10.8%)	363 (11.6%)	436 (12.1%)

Lastly, we evaluated the search efficiency on WOA09_1DEG dataset, by using almost the same parameters as the previous experiment (exception being that a fixed maximum CWE of 1.5 was used). In Table 7, we report the size of search space, the number of visited subgroups during our search, the number of visited subgroups eliminated by pruning, and the number of visited subgroups eliminated by combination. Additionally, we also report the proportion of eliminated subgroups among all the visited ones.

The size of search space is equal to the product of the number of bins over each attribute. The number of eliminated subgroups reported only includes the ones that are directly pruned or combined based on our pruning measures and combination rules, and excludes the child subgroups of those. We can make the following observations. First, the search space increases exponentially



(a) # of Subgroups on WOA09_1DEG (b) Avg. Exe Times on WOA09_1DEG (c) Avg. CWRAcc on WOA09_1DEG



(d) # of Subgroups on POP (e) Avg. Exe Times on POP (f) Avg. CWRAcc on POP

Figure 5: Evaluation on WOA09_1DEG and POP Datasets: Impact of Varying Maximum CWE and # of Bins on # of Subgroups, Execution Times, and Average CWRAcc

as more bins are used, and even with a small number of bins, the search space is still huge. This shows that a brute-force algorithm will have an unacceptable cost. However, the number of visited subgroups is at least 7 orders of magnitude smaller, and does not increase very rapidly with increasing number of bins. This is because our level-wise search strategy can eliminate unqualified or small subgroups by pruning or combination as early as possible, and hence unnecessary exploration over their child subgroups can be avoided. As a result, the depth of our search tree is relatively small, mostly less than 3. Second, we can see that around 90% of visited subgroups are pruned based on our pruning measures, and around 10% visited subgroups are combined to provide a higher quality. Therefore, generally only less than 1% of visited subgroups are finally qualified, leading to subgroups that are of most significance to the users.

7. RELATED WORK

As an important data mining technique that is often applied to data exploration and descriptive induction, subgroup discovery has been extensively studied in recent years. The existing algorithms can be broadly classified into three types: *extensions of classification algorithms* [22, 31, 34, 37, 72], *extensions of association algorithms* [7, 26, 27, 30], and *extensions of evolutionary fuzzy algorithms* [11, 14, 19]. However, only a small number of subgroup discovery algorithms are capable of processing numeric attributes. As an extension of SD-Map [7], SD-Map* is designed based on FP-growth, and it uses frequent pattern tree as the search tree. In contrast, our algorithm uses a set-enumeration tree based search strategy, with pruning and combination methods we have introduced. We have extensively compared our algorithm with SD-Map*, and have shown that our algorithm can lead to the subgroups of greater generality and higher quality, as well as up to two orders of magnitude lower execution times.

Other efforts have focused on discretization methods – for example, TargetCluster [45] discretizes target variable based on clustering, instead of the simple equi-width/equi-depth binning that is more common, whereas MergeSD [28] involves discretization with overlapping intervals, by adjusting interval bounds with a bound table. Moreover, subgroup discovery techniques have also been applied in the context of spatial databases [32, 43]. Additionally, the objective of our subgroup discovery is similar to *bump hunting* [21], which also aims to identify subsets that are considerably greater or smaller than the average of the general population. However, no existing work is designed for processing large-scale array-based scientific datasets. Compared with all the existing subgroup discovery algorithms, a key difference in our approach is that we directly operate on the compact bitmap indices, instead of the raw datasets, and utilize fast bitwise operations on them, allowing processing of larger datasets.

Reducing the relevancy or redundancy of the output is also an important issue in subgroup discovery. As a modified version of the classical subgroup discovery, *relevant subgroup discovery* is proposed to eliminate irrelevant subgroups, based on the theory of relevancy [35,36]. Relevant subgroup discovery algorithms are devised mainly based on either *optimistic estimates* or *closed sets*. Most algorithms [35, 38] based on optimistic estimates can lead to a dramatic reduction of the search space as well as the execution time, but it may not be able to guarantee correct results [25]. Closed-sets-based algorithms [23, 25] require quadratic time complexity, which is clearly not applicable to large scientific datasets. Additionally, Chen *et al.* [15] applied a sequential coverage approach where the formal relevancy criteria is not followed. Other non-redundant subgroup discovery algorithms [13, 63] proposed different redundancy criteria to follow. However, to the best of our knowledge, none of these algorithms is designed for the subgroup discovery over numeric array data. In the future, our method can be combined with a separate post-processing step of *relevancy check*.

Apart from subgroup discovery, other important descriptive data mining tasks that involve various differential analysis include contrast set mining [9], emerging pattern mining [20], and differential rule mining [40]. Again, the distinctive aspect of our algorithm is efficient processing of large-scale array data.

Although bitmap indexing was initially proposed in the context of data warehouses [46, 77, 78], recently it has been widely applied in the area of scientific data management [17, 53, 54, 61, 73]. To improve query efficiency, different binning strategies [24, 48, 55, 76] and encoding methods [33] have been proposed. Furthermore, bitmap indexing has also shown its capability of assisting various data analysis tasks [8, 52, 59, 60, 62]. In this paper, the focus has been on efficient and effective subgroup discovery based on bitmap indices.

8. CONCLUSIONS AND FUTURE WORK

This paper has presented a novel algorithm for subgroup discovery over array-based datasets. This algorithm has extended the subgroup description to processing of array data, and is capable of effectively processing numeric attributes. Moreover, our algorithm directly operates on the compact bitmap indices instead of the raw datasets, and utilizes fast bitwise operations on them, allowing processing of larger datasets. We have extensively evaluated our algorithm by using multiple real-life datasets, and compared it with a popular subgroup discovery algorithm – SD-Map*. We demonstrate both high efficiency and effectiveness of our algorithm.

In future work, we plan to extend our work in several directions. First, since the current algorithm does not ensure the relevancy among the subgroups described by different sets of attributes, we plan to design an efficient relevant subgroup discovery algorithm over scientific datasets. Second, our current algorithm can only analyze one target variable at a time – our future work will extend it to support multiple target variables. Third, the subgroup discovery results do not necessarily indicate a causal relationship between the subgroup descriptions and the high/low value of target variable, but instead, only imply an association. This can be addressed in the future. Fourth, we plan to develop a module to effectively discover exception rules for the output subgroups, where each exception rule corresponds to an interesting subset of an output subgroup, and such subset has a quality in contrast to the output subgroup it belongs to.

9. REFERENCES

[1] VIKAMINE. http://www.vikamine.org.

[2] WOA09. http://www.nodc.noaa.gov/OC5/WOA09/netcdf_data.html.

[3] WOA13. http://www.nodc.noaa.gov/OC5/woa13/woa13data.html.

[4] I. Almasri, X. Gao, and N. Fedoroff. Quick mining of isomorphic exact large patterns from large graphs. In *ICDMW*, pages 517–524. IEEE, 2014.

[5] G. Antoshenkov. Byte-aligned bitmap compression. In *DCC*, page 476. IEEE, 1995.

[6] M. Atzmüller and F. Lemmerich. Fast subgroup discovery for continuous target concepts. In *Foundations of Intelligent Systems*, pages 35–44. Springer, 2009.

[7] M. Atzmüller and F. Puppe. SD-Map–A fast algorithm for exhaustive subgroup discovery. In *Knowledge Discovery in Databases: PKDD 2006*, pages 6–17. Springer, 2006.

[8] J. Ayres, J. Flannick, J. Gehrke, and T. Yiu. Sequential pattern mining using a bitmap representation. In *SIGKDD*, pages 429–435. ACM, 2002.

[9] S. D. Bay and M. J. Pazzani. Detecting group differences: Mining contrast sets. *Data Mining and Knowledge Discovery*, 5(3):213–246, 2001.

[10] R. J. Bayardo Jr. Efficiently mining long patterns from databases. In *SIGMOD Rec.*, volume 27, pages 85–93. ACM, 1998.

[11] F. Berlanga, M. J. del Jesus, P. González, F. Herrera, and M. Mesonero. Multiobjective evolutionary induction of subgroup discovery fuzzy rules: A case study in marketing. In *Advances in Data Mining. Applications in Medicine, Web Mining, Marketing, Image and Signal Mining*, pages 337–349. Springer, 2006.

[12] A. Biswas, S. Dutta, H.-W. Shen, and J. Woodring. An information-aware framework for exploring multivariate data sets. *Visualization and Computer Graphics, IEEE Transactions on*, 19(12):2683–2692, 2013.

[13] M. Boley and H. Grosskreutz. Non-redundant subgroup discovery using a closure system. In *Machine Learning and Knowledge Discovery in Databases*, pages 179–194. Springer, 2009.

[14] C. J. Carmona, P. González, M. J. del Jesús, and F. Herrera. Non-dominated multi-objective evolutionary algorithm based on fuzzy rules extraction for subgroup discovery. In *Hybrid Artificial Intelligence Systems*, pages 573–580. Springer, 2009.

[15] H. Cheng, X. Yan, J. Han, and P. S. Yu. Direct discriminative pattern mining for effective classification. In *ICDE*, pages 169–178. IEEE, 2008.

[16] R. Chester. *Marine Geochemistry*. John Wiley & Sons, 2009.

[17] J. Chou, K. Wu, O. Rubel, M. Howison, J. Qiang, B. Austin, E. W. Bethel, R. D. Ryne, A. Shoshani, et al. Parallel index and query for large scale data analysis. In *SC*, pages 1–11. IEEE, 2011.

[18] S. Cui, V. F. de Almeida, and B. Khomami. Molecular dynamics simulations of tri-n-butyl-phosphate/n-dodecane mixture: Thermophysical properties and molecular structure. *The Journal of Physical Chemistry B*, 118(36):10750–10760, 2014.

[19] M. J. del Jesus, P. González, F. Herrera, and M. Mesonero. Evolutionary fuzzy rule induction process for subgroup discovery: A case study in marketing. *Fuzzy Systems, IEEE Transactions on*, 15(4):578–592, 2007.

[20] G. Dong and J. Li. Efficient mining of emerging patterns: Discovering trends and differences. In *SIGKDD*, pages 43–52. ACM, 1999.

[21] J. H. Friedman and N. I. Fisher. Bump hunting in high-dimensional data. *Statistics and Computing*, 9(2):123–143, 1999.

[22] D. Gamberger and N. Lavrac. Expert-guided subgroup discovery: Methodology and application. *Journal of Artificial Intelligence Research*, 17(1):501–527, 2002.

[23] G. C. Garriga, P. Kralj, and N. Lavrač. Closed sets for labeled data. *The Journal of Machine Learning Research*, 9:559–580, 2008.

[24] N. Goyal and Y. Sharma. New binning strategy for bitmap indices on high cardinality attributes. In *COMPUTE*, page 22. ACM, 2009.

[25] H. Grosskreutz and D. Paurat. Fast and memory-efficient discovery of the top-k relevant subgroups in a reduced candidate space. In *Machine Learning and Knowledge Discovery in Databases*, pages 533–548. Springer, 2011.

[26] H. Grosskreutz and S. Rüping. On subgroup discovery in numerical domains. *Data mining and knowledge discovery*, 19(2):210–226, 2009.

[27] H. Grosskreutz, S. Rüping, and S. Wrobel. Tight optimistic estimates for fast subgroup discovery. In *Machine Learning and Knowledge Discovery in Databases*, pages 440–456. Springer, 2008.

[28] F. Herrera, C. J. Carmona, P. González, and M. J. del Jesus. An overview on subgroup discovery: foundations and applications. *Knowledge and information systems*, 29(3):495–525, 2011.

[29] P. W. Jones, P. H. Worley, Y. Yoshida, J. White, and J. Levesque. Practical performance portability in the Parallel Ocean Program (POP). *Concurrency and Computation: Practice and Experience*, 17(10):1317–1327, 2005.

[30] B. Kavšek, N. Lavrač, and V. Jovanoski. Apriori-SD: Adapting Association Rule Learning to Subgroup Discovery. In *Advances in Intelligent Data Analysis V*, pages 230–241. Springer, 2003.

[31] W. Klösgen. Explora: A multipattern and multistrategy discovery assistant. In *Advances in knowledge discovery and data mining*, pages 249–271. American Association for Artificial Intelligence, 1996.

[32] W. Klösgen and M. May. Spatial subgroup mining integrated in an object-relational spatial database. In *Principles of Data Mining and Knowledge Discovery*, pages 275–286. Springer, 2002.

[33] N. Koudas. Space efficient bitmap indexing. In *CIKM*, pages 194–201. ACM, 2000.

[34] N. Lavrac, P. Flach, B. Kavsek, and L. Todorovski. Adapting classification rule induction to subgroup discovery. In *ICDM*, pages 266–273. IEEE, 2002.

[35] N. Lavrač and D. Gamberger. *Relevancy in constraint-based subgroup discovery*. Springer, 2006.

[36] N. Lavrač, D. Gamberger, and V. Jovanoski. A study of relevance for learning in deductive databases. *The Journal of Logic Programming*, 40(2):215–249, 1999.

[37] N. Lavrač, B. Kavšek, P. Flach, and L. Todorovski. Subgroup discovery with CN2-SD. *The Journal of Machine Learning Research*, 5:153–188, 2004.

[38] F. Lemmerich, M. Rohlfs, and M. Atzmueller. Fast discovery of relevant subgroup patterns. In *FLAIRS*, 2010.

[39] L. Li, J. Yang, Y. Xu, Z. Qin, and H. Zhang. Documents clustering based on max-correntropy nonnegative matrix factorization. In *ICMLC*, volume 2, pages 850–855. IEEE, 2014.

[40] T. Liu, F. Wang, J. Zhu, and G. Agrawal. Differential analysis on deep web data sources. In *ICDMW*, pages 33–40. IEEE, 2010.

[41] Z. Lv, S. U. Réhman, and G. Chen. Webvrgis: A p2p network engine for vr data and gis analysis. In *Neural Information Processing*, pages 503–510. Springer, 2013.

[42] Z. Lv and T. Su. 3d seabed modeling and visualization on ubiquitous context. In *SIGGRAPH Asia 2014 Posters*, page 33. ACM, 2014.

[43] M. May and L. Ragia. Spatial subgroup discovery applied to the analysis of vegetation data. In *Practical Aspects of Knowledge Management*, pages 49–61. Springer, 2002.

[44] R. McLay, D. James, S. Liu, J. Cazes, and W. Barth. A user-friendly approach for tuning parallel file operations. In *SC*, pages 229–236. IEEE, 2014.

[45] K. Moreland and K. Truemper. Discretization of target attributes for subgroup

discovery. In *Machine Learning and Data Mining in Pattern Recognition*, pages 44–52. Springer, 2009.

[46] P. O'Neil and D. Quass. Improved query performance with variant indexes. In *SIGMOD Rec.*, volume 26, pages 38–49. ACM, 1997.

[47] P. Riddle, R. Segal, and O. Etzioni. Representation design and brute-force induction in a Boeing manufacturing domain. *Applied Artificial Intelligence an International Journal*, 8(1):125–147, 1994.

[48] D. Rotem, K. Stockinger, and K. Wu. Optimizing candidate check costs for bitmap indices. In *CIKM*, pages 648–655. ACM, 2005.

[49] J. Shen and S. Cheung. Layer depth denoising and completion for structured-light rgb-d cameras. In *CVPR*, pages 1187–1194. IEEE, 2013.

[50] J. Shen, P.-C. Su, S.-c. S. Cheung, and J. Zhao. Virtual mirror rendering with stationary rgb-d cameras and stored 3-d background. *Image Processing, IEEE Transactions on*, 22(9):3433–3448, 2013.

[51] J. Shen and W.-t. Tan. Image-based indoor place-finder using image to plane matching. In *ICME*, pages 1–6. IEEE, 2013.

[52] S. Shohdy, Y. Su, and G. Agrawal. Accelerating data mining on incomplete datasets by bitmaps-based missing value imputation. In *DBKDA*, 2015.

[53] R. R. Sinha, S. Mitra, and M. Winslett. Bitmap indexes for large scientific data sets: A case study. In *IPDPS*, pages 10–pp. IEEE, 2006.

[54] R. R. Sinha and M. Winslett. Multi-resolution bitmap indexes for scientific data. *TODS*, 32(3):16, 2007.

[55] K. Stockinger, K. Wu, and A. Shoshani. Evaluation strategies for bitmap indices with binning. In *Database and Expert Systems Applications*, pages 120–129. Springer, 2004.

[56] H. Su and D. Zhu. An elastic mixed-criticality task model and its scheduling algorithm. In *DATE*, 2013.

[57] H. Su, D. Zhu, and D. Mossé. Scheduling algorithms for elastic mixed-criticality tasks in multicore systems. In *RTCSA*, 2013.

[58] T. Su, Z. Lv, S. Gao, X. Li, and H. Lv. 3d seabed: 3d modeling and visualization platform for the seabed. In *ICMEW*, pages 1–6. IEEE, 2014.

[59] Y. Su, G. Agrawal, J. Woodring, K. Myers, J. Wendelberger, and J. Ahrens. Taming massive distributed datasets: data sampling using bitmap indices. In *HPDC*, pages 13–24. ACM, 2013.

[60] Y. Su, Y. Wang, and G. Agrawal. In-situ bitmaps generation and efficient data analysis based on bitmaps. In *HPDC*. ACM, 2015.

[61] Y. Su, Y. Wang, G. Agrawal, and R. Kettimuthu. SDQuery DSI: integrating data management support with a wide area data transfer protocol. In *SC*, page 47. ACM, 2013.

[62] T. Uno, M. Kiyomi, and H. Arimura. Lcm ver. 3: Collaboration of array, bitmap and prefix tree for frequent itemset mining. In *OSDM*, pages 77–86. ACM, 2005.

[63] M. van Leeuwen and A. Knobbe. Non-redundant subgroup discovery in large and complex data. In *Machine Learning and Knowledge Discovery in Databases*, pages 459–474. Springer, 2011.

[64] J. Wang, N. Abu-Ghazaleh, and D. Ponomarev. Interference Resilient PDES on Multi-core Systems: Towards Proportional Slowdown. In *SIGSIM-PADS*, pages 115–126, 2013.

[65] J. Wang, K. Bahulkar, D. Ponomarev, and N. Abu-Ghazaleh. Can PDES Scale in Environments with Heterogeneous Delays? In *SIGSIM-PADS*, pages 35–46, 2013.

[66] J. Wang, D. Jagtap, N. Abu-Ghazaleh, and D. Ponomarev. Parallel discrete event simulation for multi-core systems: Analysis and optimization. *IEEE Trans. Parallel Distrib. Syst.*, 25(6):1574–1584, 2014.

[67] J. J.-Y. Wang, Y. Wang, S. Zhao, and X. Gao. Maximum mutual information regularized classification. *Engineering Applications of Artificial Intelligence*, 37:1–8, 2015.

[68] Y. Wang, G. Agrawal, G. Ozer, and K. Huang. Removing sequential bottlenecks in analysis of next-generation sequencing data. In *IPDPSW*, pages 508–517. IEEE, 2014.

[69] Y. Wang, A. Nandi, and G. Agrawal. SAGA: array storage as a DB with support for structural aggregations. In *SSDBM*, page 9. ACM, 2014.

[70] Y. Wang, Y. Su, and G. Agrawal. Supporting a Light-Weight Data Management Layer Over HDF5. In *CCGRID*, pages 335–342, may 2013.

[71] Y. Wang, J. Wei, and G. Agrawal. SciMATE: A Novel MapReduce-Like Framework for Multiple Scientific Data Formats. In *CCGRID*, pages 443–450, may 2012.

[72] S. Wrobel. An algorithm for multi-relational discovery of subgroups. In *Principles of Data Mining and Knowledge Discovery*, pages 78–87. Springer, 1997.

[73] K. Wu, W. Koegler, J. Chen, and A. Shoshani. Using bitmap index for interactive exploration of large datasets. In *SSDBM*, pages 65–74. IEEE, 2003.

[74] K. Wu, E. J. Otoo, and A. Shoshani. Compressing bitmap indexes for faster search operations. In *SSDBM*, pages 99–108. IEEE, 2002.

[75] K. Wu, K. Stockinger, and A. Shoshani. Breaking the curse of cardinality on bitmap indexes. In *SSDBM*, pages 348–365. Springer, 2008.

[76] K.-L. Wu and P. S. Yu. Range-based bitmap indexing for high cardinality attributes with skew. In *COMPSAC*, pages 61–66. IEEE, 1998.

[77] M.-C. Wu. Query optimization for selections using bitmaps. In *SIGMOD Rec.*, volume 28, pages 227–238. ACM, 1999.

[78] M.-C. Wu and A. P. Buchmann. Encoded bitmap indexing for data warehouses. In *ICDE*, pages 220–230. IEEE, 1998.

[79] B. Xie, Y. Mu, M. Song, and D. Tao. Random projection tree and multiview embedding for large-scale image retrieval. In *Neural Information Processing. Models and Applications*, pages 641–649. Springer, 2010.

[80] D. Yang, E. A. Rundensteiner, and M. O. Ward. Analysis guided visual exploration of multivariate data. In *VAST*, pages 83–90. IEEE, 2007.

[81] Z.-H. You, S. Li, X. Gao, X. Luo, and Z. Ji. Large-scale protein-protein interactions detection by integrating big biosensing data with computational model. *BioMed research international*, 2014, 2014.

[82] F. Železnỳ and N. Lavrač. Propositionalization-based relational subgroup discovery with RSD. *Machine Learning*, 62(1-2):33–63, 2006.

[83] S. Zhang, D. Caragea, and X. Ou. An empirical study on using the national vulnerability database to predict software vulnerabilities. In *Database and Expert Systems Applications*, pages 217–231. Springer, 2011.

[84] Y. Zhou, L. Li, T. Zhao, and H. Zhang. Region-based high-level semantics extraction with cedd. In *IC-NIDC*, pages 404–408. IEEE, 2010.