Discovering Coherent Value Bicliques In Genetic Interaction Data

Gowtham Atluri
Dept of Comp Sc and Engg
Univ of Minnesota, Twin Cities
Minneapolis, MN USA
gowtham@cs.umn.edu

Jeremy Bellay
Dept of Comp Sc and Engg
Univ of Minnesota, Twin Cities
Minneapolis, MN USA
bellay@cs.umn.edu

Gaurav Pandey
Dept of Comp Sc and Engg
Univ of Minnesota, Twin Cities
Minneapolis, MN USA
gaurav@cs.umn.edu

Chad Myers
Dept of Comp Sc and Engg
Univ of Minnesota, Twin Cities
Minneapolis, MN USA
cmyers@cs.umn.edu

Vipin Kumar
Dept of Comp Sc and Engg
Univ of Minnesota, Twin Cities
Minneapolis, MN USA
kumar@cs.umn.edu

ABSTRACT

Genetic Interaction (GI) data provides a means for exploring the structure and function of pathways in a cell. Coherent value bicliques (submatrices) in GI data represents functionally similar gene modules or protein complexes. However, no systematic approach has been proposed for exhaustively enumerating all coherent value submatrices in such data sets, which is the problem addressed in this paper. Using a monotonic range measure to capture the coherence of values in a submatrix of an input data matrix, we propose a two-step Apriori-based algorithm for discovering all nearly constant value submatrices, referred to as Range Constrained Blocks. By systematic evaluation on an extensive genetic interaction data set, we show that the coherent value submatrices represent groups of genes that are functionally related than the submatrices with diverse values. We also show that our approach can exhaustively find all the submatrices with a range less than a given threshold, while the other competing approaches can not find all such submatrices.

1. INTRODUCTION

Genetic Interaction (GI) data provides a means for exploring the structure and function of pathways in a cell [18]. The development of technologies like Synthetic Genetic Array (SGA) and Epistatic MiniArray (E-MAP), have enabled large-scale measurement of quantitative interactions in *S. Cerevisiae* [20]. These technologies measure the interaction between two genes in terms of the fitness of a cell when a pair of genes are knocked out relative to the expected fitness when there is no interaction between the pair of genes. Specifically, two genes A and B are said to interact genetically if the fitness of a large set of yeast cells (colony) after the deletion of both genes (say F_{AB}) differs from the expected fitness if the effects of A and B were independent, i.e., the product of the fitnesses after the deletion of A (say F_A) and B (say F_B) individually [20]. Thus two genes interact if $\epsilon \neq 0$ in the following equation.

$$\epsilon = F_{AB} - F_A F_B \tag{1}$$

The magnitude of this score, i.e., $|\epsilon|$ represents the strength of the genetic interaction between A and B. In addition, if $\epsilon > 0$, the interaction is called a “positive” or “alleviating” interaction, and $\epsilon < 0$ denotes a “negative” or “aggravating” interaction. A GI interaction data set can be represented as an adjacency matrix G, where the value of each element G_{ij} is the interaction score between the query gene g_i (row) and the array gene g_j (column), calculated using Equation 1.

Previous studies on analyzing genetic interaction networks has noted striking structure present in these networks. For example, [13, 21] have noted the presence of nearly complete bipartite subgraphs involving similar type of interactions. The two sets of genes in each of the bipartite subgraphs typically represent pairs of functionally complementary pathways or protein complexes. Previous efforts to discover these bipartite subgraphs in GI data are limited to finding bipartite subgraphs with interactions having same sign [13, 21] i.e., they look for bicliques such that all interactions are positive (or all interactions are negative) without being concerned about the variation in the magnitude of the interactions. It has been observed that bicliques with coherent (i.e., similar values) positive interaction scores represent protein complexes or modules of genes involved in similar biological functions [3, 18]. In this paper we address the problem of discovering such bicliques i.e. submatrices with coherent values in a GI data matrix.

This problem of discovering a submatrix with coherent values is very similar in nature to the biclustering problem ([15]) that is addressed in the domain of micorarray data analysis. In biclustering, the goal is to find a subset of the gene (rows) constituting a gene expression data set that have coherent values across a subset of the conditions (columns). Several algorithms have been proposed in the literature for finding such biclusters. These algorithms vary in their definition of “coherence”, and thus focus on different types of

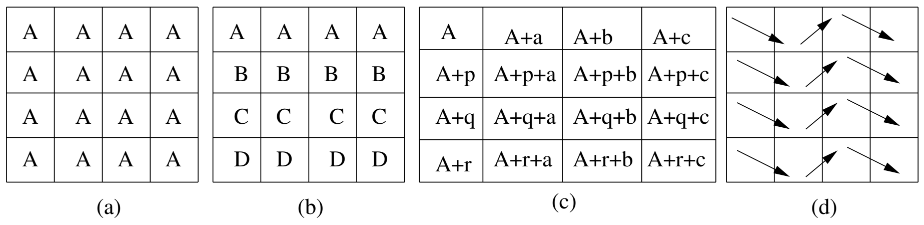

Figure 1: Types of biclusters: (a) Biclusters with constant values (b) Biclusters with constant rows (c) Biclusters following an additive model (d) Biclusters with coherent evolutions.

biclusters corresponding to this definition. [15] have classified the biclusters found by these algorithms into four categories, as shown in Figure 1. These categories include (i) biclusters with constant values, (ii) biclusters with constant rows or columns, (iii) biclusters following an additive (or multiplicative) model, and (iv) biclusters with coherent evolutions. The problem we address in this paper is the same as “finding constant value biclusters” as defined in [15].

Several biclustering algorithms, such as CC ([9]), ISA ([6]), SAMBA ([19]), OPSM ([5]) and co-clustering ([10]) , have been proposed to find different types of these biclusters. However, these approaches suffer from three common limitations. (i) Most of these approaches either adopt top-down greedy schemes that start from all rows and columns, and then iteratively eliminate rows and/or columns to optimize their objective function ([10, 9]), or start with a random initial seed and use heuristics to converge to the final bicluster ([6, 5]). Due to the use of these heuristics, these algorithm are unable to search the space of all possible biclusters exhaustively. (ii) The objective of these approaches is different from finding coherent value submatrices. For example, CC finds constant row biclusters which have low mean squared residue score and SAMBA finds maximum weight bicliques. (iii) Small biclusters tend to get overshadowed by noise and/or by larger biclusters due to the top-down nature of the search. In particular, these techniques are not meant to find constant value biclusters that are of interest to us.

Interestingly, pattern mining algorithms developed in association analysis ([2, 8, 11]) also produce biclusters in binary market-basket-type data, where each row is a transaction that indicates the purchase of items (represented along columns) in a store. A pattern is a group of items (itemset) purchased together in atleast a given fraction of transactions and it can be represented as a submatrix with supporting transactions as rows and items in the itemset as columns, with all the values included being 1. So, these patterns are essentially similar to constant value biclusters that we seek to discover. However, they only work with binary data sets. Recently, [16] have extended these algorithms to find constant row/column biclusters in real-valued data, but their approach still can not discover constant value biclusters exclusively. Although constant row biclusters may include constant value biclusters, these need to be identified by post-processing, as we discussed in the evaluation section, this is not an effective way to find coherent value biclusters.

In this paper, we present a novel framework to exhaustively discover all RCBs in a given GI dataset. For this, we define the notion of a coherent submatrix whose values are within a pre-specified (relative) range, and refer to it as a *Range Constrained Block* (RCB). The measure of coherence used, named the *Range* measure, is monotonic in nature, and thus makes it possible to develop an Apriori-like algorithm ([1, 2]) to enumerate all RCBs whose value for the *Range* measure is lower than the user-specified threshold. This algorithm is guaranteed to recover all such coherent submatrices in the given data set.

The rest of the paper is organized as follows. We discuss some related approaches for the bicluster discovery problem in Section 2. We present the RCB discovery framework in Section 3. Section 4 details the quantitative evaluation of RCBs. We conclude with suggestions for future work in Section 5.

2. RELATED APPROACHES

Although our work is the first systematic approach for the problem of finding constant value biclusters, this problem can be approached in other ways also. One of the most straightforward approaches would be to binarize the data matrix and use the Apriori algorithm [1] to find binary frequent patterns, which are also biclusters. However, this is not an ideal approach for our problem, since all the values are represented by 1 or 0, and thus even if such a pattern is found, there is no guarantee on the coherence of the values included in a bicluster so found. This problem is shared by Ma *et al.*'s approach [14] for finding highly connected subgraphs from a bipartite graph representation of GI data. Ma *et al.*'s approach further faces the problem of being non-exhaustive due to the heuristic search algorithm employed. Another possible approach is to generate multiple binary matrices with each matrix having 1s for values that are within in a small range (window). This approach can not find biclusters that have values that are in two adjacent windows but still in a small range. Below, we discuss three related approaches that focus on finding biclusters directly from real-valued data. Note that these methods were originally developed for microarray data, but the formulations and underlying principles apply directly to other types of data also.

2.1 Range Support Patterns (RAP)

Pandey *et al.* [16] recently proposed an association analysis approach for finding constant row/column biclusters (Figure 1(b)) directly from real-valued data. Here, they defined the *RangeSupport* measure of an itemset as the sum of the contributions of each transaction where the values of these items are within a pre-specified (relative) range threshold α, and are of the same sign. This contribution is defined to be the minimum of the absolute value among the items for a transaction that satisfies both these conditions, and zero otherwise. This definition makes *RangeSupport* anti-monotonic, and an Apriori-like algorithm is then used to mine constant row/column biclusters from the given data set. This approach has several desirable properties, such as the exhaustive enumeration of all biclusters of this type, the possibility of overlaps between biclusters and the ability to discover small biologically meaningful biclusters. However, these biclusters are only guaranteed to be coherent over one of the dimensions (row or column), but not necessarily both the dimensions, as is required for constant value biclusters.

2.2 Cheng and Church's algorithm

Cheng and Church [9] (CC) proposed the first algorithm, which we refer to as CC, to find biclusters in microarray data. They used the mean squared residue (MSR) measure to capture the coherence of expression values among a set of genes across a subset of all the conditions, and focused on finding biclusters with low MSR values. However, since enumerating all such biclusters is an NP-hard problem, a greedy heuristic approach to discover such biclusters is used. This approach first starts with the entire matrix M and iteratively removes rows or columns that provided maximum reduction in the MSR score until the MSR score is below a user specified threshold, or a certain number of iterations is reached. Provisions are also made for finding overlapping biclusters. However, this algorithm faces several challenges in finding constant value biclusters. First, since a heuristic search algorithm is employed, it can not be guaranteed that all biclusters with an MSR lower than the specified threshold will be found. Also, CC generally finds biclusters of large sizes since the termination criteria are generally satisfied early in the search process. Finally, CC tends to find several biclusters with almost neutral (zero) values in them, since they have MSR=0, which may not be useful if these biclusters need to be analyzed further.

2.3 SAMBA

Tanay *et al.* [19] proposed the SAMBA algorithm for finding biclusters, which they define as a group of genes that jointly respond to a group of conditions. A gene is said to respond to a condition if its expression level changes significantly relative to its expression under normal conditions. The given gene expression data matrix is represented as a bipartite graph with genes and conditions as the two sets of vertices. An edge e connects gene u to condition v with weight 1 if the expression level of u is significant under v and -1 otherwise. The algorithm then tries to find maximal weight subgraphs, all of whose edges of the same sign, in this weighted bipartite graph using a heuristic search algorithm. The genes and conditions constituting these subgraphs are output as biclusters. It can be seen that, similar to binary pattern mining, SAMBA ignores the importance of the real values once it is determined if a value is significant or not. Thus, the coherence of values constituting the resultant biclusters is not guaranteed. Furthermore, SAMBA can not guarantee finding all possible maximal weight subgraphs, which is an NP-hard problem.

In summary, although various algorithms have been proposed for finding different types of biclusters, none of them exhaustively finds constant value biclusters, which are the focus of our work. The challenges faced by these approaches for this problem are reflected in the experimental results discussed in Section 4.

3. RCB DISCOVERY APPROACH

In this section we introduce an Apriori-like framework to mine RCBs from a real valued data set. For this, we first define a *range* measure to capture the semantics of an RCB and prove that it is monotonic. We then introduce a diagonal representation of a square sub-matrix, and describe how it can be used to efficiently mine square RCBs using an Apriori-like algorithm. This algorithm discovers a rectangular RCB in the form of multiple, overlapping square RCBs. Finally, we present an Apriori-like algorithm to merge these square RCBs, at the end of each level in the previous algorithm, into rectangular RCBs. Note that although the RCB mining framework is defined below for a data matrix that has items of the same type on both of its dimensions, it can be also be used for a data set that has different types of items along the two dimensions.

3.1 Range measure

We defined RCB as a submatrix that has all values within a given range. This range can simply be defined as a difference between the maximum and minimum value of the submatrix. However, since most real data sets have a wide range of values, we use a relative form of range to make its definition more versatile. Formally, if G is any real valued positive data matrix, for any submatrix G_{IJ}, where $I = i_1, i_2, \ldots, i_k$ and $J = j_1, j_2, \ldots, j_l$ constitute its two dimensions, and whose each element is g_{ij} ($i \in I$ and $j \in J$), the range of G_{IJ} is defined in a straight-forward manner as:

$$range(G_{IJ}) = \frac{\max_{i\in I, j\in J}(g_{ij}) - \min_{i\in I, j\in J}(g_{ij})}{\min_{i\in I, j\in J}(g_{ij})} \qquad (2)$$

However, another complicating aspect of real-valued data sets is that they contain both positive and negative values. For example, in genetic interaction data, positive and negative values represent different types of interactions, as discussed earlier. This factor needs to be incorporated into the definition of *range*, so that the resultant RCBs are coherent not only in values, but also in their signs. We ensure this by enforcing this constraint into the definition of the *range* measure as formulated in Equation 3. Here, the range of a submatrix that includes both positive and negative values is simply set to infinity, so that it is not considered as an RCB. Note that this constraint is supported by research on genetic interactions, where it has been shown that groups of genes (modules) having interactions of same type (sign) are more functionally related than those involved in very different types of interactions [21].

$$r(G_{IJ}) = \begin{cases} range(abs(G_{IJ})) \\ \qquad\qquad (if\ g_{ij} > 0 \ \ \forall\ i \in I,\ \forall\ j \in J \\ \qquad\qquad\qquad\qquad or \\ \qquad\qquad g_{ij} < 0 \ \ \forall\ i \in I,\ \forall\ j \in J) \\ \infty \quad (otherwise) \end{cases} \qquad (3)$$

Using this definition, it can be shown that the *range* measure has a monotonicity property, as shown by the following.

THEOREM 1. *Range measure is monotonic*

PROOF. Consider a submatrix G_{IJ} of a matrix G, where $I = i_1, i_2, \ldots, i_k$ and $J = j_1, j_2, \ldots, j_l$ are the two dimensions of the submatrix and $r(G_{IJ}) \in [0, \infty)$. Let $I' = I \cup i_{k+1}$ and $J' = J \cup j_{l+1}$.

The range of the submatrix $r(G_{I'J'})$ will fall into one of the following:

• The elements in $G_{I'J'}$ have different signs: Now, $r(G_{I'J'}) =$

∞. Since $r(G_{IJ}) \in [0, \infty)$, $r(G_{I'J'}) \geq r(G_{IJ})$.
• The elements in $G_{I'J'}$ have the same sign: Two sub-cases are possible in this scenario:
 $- \ max(G_{I'J'}) = max(G_{IJ})$ and $min(G_{I'J'}) = min(G_{IJ})$. Then $r(G_{I'J'}) = r(G_{IJ})$.
 $- \ max(G_{I'J'}) \geq max(G_{IJ})$ and/or $min(G_{I'J'}) \leq min(G_{IJ})$. Then $r(G_{I'J'}) \geq r(G_{IJ})$.
Thus, $r(G_{IJ})$ is monotonic. $\square$

Due to this monotonicity property, the *range* measure can be used in a bottom-up Apriori-like algorithm to enumerate the all the RCBs in a given data matrix that satisfy the given range constraint. Note that traditional frequent pattern mining algorithms focus on patterns with support greater than a user-specified threshold, while we discover RCBs with range lower than the user-specified threshold, thus enabling us to ensure coherence in both the dimensions simultaneously. However, due to the complexities in this search process discussed below, we adopt a two-step process, in which first all the square submatrices that qualify to be an RCB are enumerated, and then, these square RCBs are merged to form rectangular RCBs of arbitrary sizes. We describe the individual components of this process below.

3.2 Challenges in finding RCB patterns using the standard Apriori like approaches

This process of finding RCBs, a search in a combination of two dimensions, is a non-trivial problem and is computationally hard compared to the problem of frequent itemset discovery. In discovering frequent itemsets, the Apriori algorithm starts with a single items that are frequent. These individual items are then merged to form candidate size-2 itemsets and their supported is computed. All frequent pairs are further merged to form candidate itemsets of size-3 and their support is computed. This process is repeated until no more bigger itemsets can be found. One can design a similar approach for finding RCBs that hold a range constraint (r) in a given genetic interaction matrix with m rows and n columns as follows: all $m \times n$ individual elements in the matrix are considered as candidate size-1×1 RCBs. Each element that has a non-zero value is considered as a size-1×1 RCB, because range (r) for zero valued elements is ∞. Now, for each of the size-1×1 RCBs a row or column is added to form candidate size-1×2 (or candidate size-2×1) RCBs. The range measure can then be computed to determine size-1×2 (or size-2×1) RCBs. This approach for finding an $m \times n$ RCB involves enumeration of $(2^m - 1)(2^n - 1)$ smaller sub-matrices in the process of discovering it. Where as, finding a size-n itemset involves enumerating $(2^n - 1)$ smaller itemsets.

Thus, RCB discovery is a combinatorial search in $m \times n$ space, whereas traditional frequent pattern mining is a search in n-dimensional space. As a result searching for RCBs in matrix with large dimensionality can be computationally inefficient if done in a simplistic fashion. In the following subsection we present an approach to represent a square RCB in the form a one-dimensional vector which helps in discovering square RCBs efficiently.

3.3 Diagonal representation of square RCBs

We make use of the observation that a square sub-matrix can be represented by the indices along the diagonal. Consider a square sub-matrix G_{IJ}, where $I = i_1, i_2, \ldots, i_k$ and $J = j_1, j_2, \ldots, j_k$ are its two dimensions. This sub-matrix is can be represented by its diagonal $\{(i_1, j_1), (i_2, j_2), \ldots (i_k, j_k)\}$. In other words, we can write it as $\{d_{i_1 j_1}, d_{i_2 j_2}, \ldots d_{i_k j_k}\}$, where $d_{i_m, j_n} = (i_m, j_n)$ for $\forall i_m \in I, \ j_n \in J$. This *diagonal-set* for a matrix can be considered analogous to an *itemset* in the traditional association analysis. This representation makes it easier to represent a size k square sub-matrix as a one-dimensional vector of pairs of indices of length k. Using this representation, all square RCBs can now be enumerated efficiently in a manner similar to discovering frequent itemsets in binary datasets. Thus the diagonal representation facilitates efficient Apriori-based search for RCBs by mapping the two-dimensional search space into one-dimensional search space.

In the following sub-section we present an Apriori like algorithm that makes use of the diagonal representation for discovering square RCBs. Since RCBs can be rectangular, we then present an efficient algorithm that merges the square RCBs of same size into rectangular RCBs in an Apriori-like fashion.

3.4 Mining Square RCBs

As the first step of our RCB discovery process, we use the following Apriori-like algorithm to discover all square RCBs in a given data matrix for a user-specified range threshold.

Algorithm 1: 2-D Square RCB Approach

```
Input:
i. $G$, a real valued data matrix of size $|m \times n|$, with items
   $I = \{i_1, i_2, \ldots i_m\}$ and $J = \{j_1, j_2, \ldots j_n\}$ along the two dimensions
ii. $\delta$, a range threshold
Output:
All square submatrices $G_{I'J'}$ in $G$ with $r(G_{I'J'}) \leq \delta$

$k = 1$
$S_k = \{d_{ij} | g_{ij} \neq 0\}$    // Find all size $|1 \times 1|$ RCBs
while $F_k \neq \emptyset$ do
  $k = k + 1$
  $CS_k = Apriori - gen(S_{k-1})$
                              // Generate all size $k$ candidate RCBs
  for each candidate $cs_k \in CS_k$ do
      compute $r(cs_k)$ using Eq. 3
  end
  $S_k = \{cs_k | cs_k \in CS_k \wedge r(cs_k) \leq \delta\}$
end
Result = $\bigcup S_k$
```

This algorithm takes a real valued data matrix and a user-specified range threshold as input and enumerates all square sub-matrices in the given matrix for which the range constraint holds. To begin, since the *range* measure defined in Equation 3 is ∞ for any sub-matrix that has all zero elements, each non-zero element in the given data matrix is treated as a level-1 square RCB. At level-2, each level-1 RCB is paired with another level-1 RCB that has both indices greater than itself to form candidate level-2 square RCBs. Now, all the candidates that satisfy the range constraint are output as level-2 square RCBs. At the next level, a candidate level-3 square RCB is generated from two level-2 square RCBs using $Apriori - gen$ [2], a method used to efficiently generate candidate level-k itemsets from level-$k-1$ frequent itemsets. $Apriori - gen$ constructs a new candidate level-3 square RCB by combining two level-2 RCBs if they have one element of the diagonal-set in common. More generally, a candidate level-$(k+1)$ square RCB is generated from two level-k square RCBs if their diagonal-sets overlap in $k-1$ elements. Let $\{d_{i_1j_1},\ d_{i_2j_2},\ \dots\ d_{i_{k-1}j_{k-1}},\ d_{i_kj_k}\}$ and $\{d_{i_1j_1},\ d_{i_2j_2},\ \dots\ d_{i_{k-1}j_{k-1}},\ d_{i_{k+1}j_{k+1}}\}$ be two level-k square RCBs. Then, a level-$(k+1)$ candidate square RCB is obtained by merging these two level-k square RCBs that have $k-1$ elements of their diagonal-sets overlapping. Finally, the candidate RCBs that satisfy the range constraint are enumerated as the level-$(k+1)$ square RCBs. This process is continued until no more sub-matrices satisfy the range constraint.

3.5 Combining Square RCBs into Rectangular RCBs

Algorithm 1 can be used to discover all square RCBs in a given data matrix. However, a naturally existing rectangular RCB of size $m \times n$ ($m > n$) will result in $\binom{m}{n}$ square RCBs that share the same n indices along the shorter dimension. As each rectangular sub-matrix is broken into multiple square sub-matrices that share the shorter dimension, we need a method to join them. It is important to note that all the squares that share the same dimension may not form a rectangular RCB, but some combinations of these squares could potentially hold the range constraint to form rectangular RCBs. So, we use the following Apriori-like algorithm to combine these square RCBs into rectangular ones that satisfy the range constraint.

Algorithm 2: MINING RECTANGULAR RCBS

Input:
i. G, a real valued data matrix of size $|m \times n|$, with items $I = \{i_1, i_2, \dots i_m\}$ and $J = \{j_1, j_2, \dots j_n\}$ along the two dimensions
ii. δ, a range threshold
iii. All maximal square RCBs at level-l that are enumerated by Algorithm 1
Output:
All rectangular submatrices $G_{I'J'}$ whose smallest dimension is of length k in G with $r(G_{I'J'}) \le \delta$

```
for each set of square RCBs $S = \{s_1, s_2, \dots s_t\}$ that have one dimension of the square common do
    $k = 1$
    $R_k = \{s_i | \forall s_i \in S\}$
    while $R_k \neq \emptyset$ do
        $k = k + 1$
        $CR_k = Apriori - gen(R_{k-1})$
            // Generate all size $k$ candidate rectangular RCBs
        for each candidate $cr_k \in CR_k$ begin
            compute $r(cr_k)$ using Eq. 3
        end
        $R_k = \{cr_k | cr_k \in CR_k \wedge r(cr_k) \le \delta\}$
    end
    Result = $\bigcup R_k$
end
```

Algorithm 2 takes all the maximal square RCBs at level-l of Algorithm 1 as input and for each group of square RCBs that have the same set of items across one dimension, it first considers these square RCBs as level-1 rectangular RCBs, analogous to level-1 itemsets in Apriori. It then enumerates all possible combinations of level-1 rectangular RCBs using $Apriori - gen$. Let $s_{(S,T_i)}$ and $s_{(S,T_j)}$ be the square RCBs that have one dimension (S) in common and the other dimension ($T_i, T_j \in T$) that is different. A candidate level-2 rectangular RCB is obtained by merging the dimension that is not common ($T_i \cup T_j$) and by retaining the common dimension (S). So, the set of level-2 candidates is represented as $cr_{(S,(T_i \cup T_j))}$. The candidates that satisfy the range threshold are treated as level-2 rectangular RCBs R_2. Similarly, at any level-k, $Apriori - gen$ is used to find candidate level-k rectangular RCBs CR_k from level $k-1$ rectangular RCBs R_{k-1}. All candidates that satisfy the range constraint are enumerated as level-k rectangular RCBs R_k. This process is iterated until no more candidate rectangular RCBs satisfy the range threshold.

Thus, using Algorithm 2 all arbitrary size RCBs are enumerated that satisfy the user-specified range constraint. The correctness of the overall RCB discovery algorithm is ensured, since only the candidates that pass the range threshold are returned as RCBs. Theorem 2 proves the completeness of this algorithm.

THEOREM 2. *RCB approach discovers all valid RCBs at a given range r in a given data set G.*

PROOF. We prove this by induction. We first prove that all valid square RCBs will be discovered by Algorithm 1. Let G be the input data matrix. In the first level, all non-zero elements in G are considered as level-1 square RCBs, since the range of a non-zero element is zero and that of a zero element is ∞. So, level-1 is complete. Now, consider the set of level-k square RCBs S_k and assume that it is complete. Since, Algorithm 1 uses $Apriori - gen(S_k)$ to enumerate all possible candidate level-$(k+1)$ square RCBs CS_{k+1} and tests them for the range constraint, level-$(k+1)$ is also complete. By induction, Algorithm 1 generates the complete set of square RCBs at any level.

We now prove that Algorithm 2 generates all possible rectangular RCBs from the set of level-l square RCBs. Since Algorithm 2 finds rectangular RCBs from each group of level-l square RCBs that share one dimension, we focus on proving that Algorithm 2 generates all possible rectangular RCBs for one such group. Since Algorithm 2 considers all level-l square RCBs that have one dimension in common, the set of level-1 rectangular RCBs R_1 is complete. Now, consider level-k rectangular RCBs R_k and assume that it is complete. Since $Apriori - gen(R_k)$ is used to generate all candidate level-$(k+1)$ rectangular RCBs CR_{k+1} and each candidate in CR_{k+1} is tested for the range constraint, level-$(k+1)$ in Algorithm 2 is complete. By induction, Algorithm 2 generates the complete set of rectangular RCBs at any level.

This proves the completeness of our algorithm. □

We now discuss the performance of this RCB discovery algorithm when tested on genetic interaction data.

4. EXPERIMENTAL RESULTS

In this section we evaluate the proposed RCB discovery approach on a genetic interaction data set and compare its performance with other approaches discussed in Section 2, namely binary frequent patterns(FP), RAP, CC and SAMBA.

4.1 Experimental Design

We tested our proposed scheme on a dataset of genetic interactions consisting of weighted positive interactions, among 500 query (row) and 3893 array (column) genes. The values in this data set belong to the interval $[0, 357]$. As the values close to zero correspond to neutral interactions and are also prone to distortion due to noise, we used a sparsification threshold γ, and replaced the values less than γ by zero. Sparsification threshold used in our experiment vary between 30 and 50. Sparsified matrices are used for mining maximal RCB biclusters at varying range thresholds between 0.3 and 1.5 and different aspects of the performance of the RCB mining approach are analyzed. Note that only RCBs larger than 3×3 are analyzed to simplify the discussion since there are far too many blocks of smaller sizes (many of which could be spurious due to noise in the data).

To assess the relative utility of our proposed scheme with respect to the approaches discussed in Section 2, namely binary frequent patterns(FP), RAP, CC and SAMBA, we also generated biclusters using them. Binary frequent patterns are generated by first constructing a binary matrix $G_{1/0}$ from the data matrix G, where each element $g_{1/0_{ij}}$ is 1 if its corresponding element in G, $g_{ij} \geq \gamma$, and 0 otherwise. Borgelt's implementation [7] of Apriori algorithm [2] is used to discover frequent patterns (biclusters) from the binary matrix $G_{1/0}$. The lowest possible support threshold with which this implementation could run without 'running out of memory' was chosen at different sparsification thresolds, and is reported in Table 1 as σ. The RAP code[1] is then used to discover biclusters on the matrix G using support thresholds determined using the median support of each item. Since RAP is meant to find one-dimensional constant row biclusters (Figure 1(b)), we used it to discover biclusters from both the dimensions (array and query genes respectively) of G, in order to ensure completeness for comparison. The RAP and binary patterns (biclusters) obtained from these transformed matrices are filtered out if the length of any one dimension is less than 3, since we limit our analysis to RCB biclusters of size 3×3 or more. Note that only closed patterns obtained from Apriori and RAP patterns are considered for further analysis, since they represent all distinct patterns. Note that maximal patterns found by RCB (since they are computed in two dimensions) correspond to closed itemsets. In addition, we also generated biclusters from this data set sparsified using $\gamma = 40$ using the SAMBA algorithm implemented in the Expander tool [17] with the parameter *#probes* set to 10 and 100. Finally, we also generated 100 biclusters using CC with two parameter settings $\delta = 0.3$ and $\delta = 0.5$ as specified in the BiCAT tool [4]. CC when run on a sparsified version of the data discovered biclusters filled with zeros, owing to the fact that MSE for a bicluster with zeros is zero. So, to help CC find reasonbale biclusters the original version of the data matrix was used. All these experiments are run on an eight-processor computer with total 32 GB memory, running Linux.

4.2 Quantitative Evaluation of RCBs

Table 1 provides a global overview of the performance of the different algorithms. In this subsection, we discuss some of these aspects in detail. This table presents the different parameter settings, number of biclusters found, variation in the size of biclusters (denoted as $L - M$, where $L = |I| \times |J|$ for the smallest bicluster G_{IJ} and M is computed similarly for the largest bicluster) and range (r as in Eq 3).

From Table 1 it can be seen that the number of RCBs, FP patterns discovered at low sparsification threshold γ are more in number due to the density of the values in the matrix. As the sparsification threshold is increased, RCB, FP and RAP discovery approaches discover fewer patterns. It is important to note that FP and RAP approaches use support to contain the complexity of the search space, where as RCB uses range measure to contain the complexity. Since no coherence in values is ensured on FP and RAP patterns they have high average range, where as the patterns discovered by RCB patterns have the average range less than the specified thresholds. The CC and SAMBA patterns are generally very large in size because of the top-down approach that is employed and also contain many 0 values within them. So, we do not consider these for further analysis.

In summary, our RCB mining approach is able to discover a larger number of coherent blocks from the genetic interaction matrix, which have low average range compared to the competing approaches. Furthermore, RCB biclusters also cover more interactions in a GI dataset as shown in Section 4.5.

Title	Parameter Settings	# biclusters	Size of biclusters	Average Range (r)
RCB biclusters				
RCB1	$\delta = 0.3,\ \gamma = 30$	8794	9-18	0.2617
RCB2	$\delta = 0.5,\ \gamma = 30$	107799	9-27	0.4352
RCB3	$\delta = 0.3,\ \gamma = 40$	991	9-15	0.2600
RCB4	$\delta = 0.5,\ \gamma = 40$	12099	9-24	0.4333
RCB5	$\delta = 0.7,\ \gamma = 40$	44044	9-39	0.6041
RCB6	$\delta = 1,\ \gamma = 40$	127884	9-51	0.8541
RCB7	$\delta = 0.7,\ \gamma = 50$	6847	9-30	0.5920
RCB8	$\delta = 1,\ \gamma = 50$	16182	9-45	0.8237
RCB9	$\delta = 1.2,\ \gamma = 50$	24584	9-48	0.9840
RCB10	$\delta = 1.5,\ \gamma = 50$	32049	9-57	1.1722
Binary Patterns				
FP1	$\sigma = 6,\ \gamma = 30$	581916	30-102	4.6778
FP2	$\sigma = 4,\ \gamma = 40$	142480	12-92	2.3624
FP3	$\sigma = 3,\ \gamma = 50$	30663	9-75	2.1087
RAP biclusters (on query genes)				
RAP1	$\sigma = 500,\ \delta = 0.5$	146	15-45	1.8999
RAP2	$\sigma = 500,\ \delta = 0.7$	610	15-72	1.9673
RAP3	$\sigma = 500,\ \delta = 1$	1758	15-93	2.1386
RAP biclusters (on array genes)				
RAP4	$\sigma = 257,\ \delta = 0.5$	2662	9-27	1.5870
RAP5	$\sigma = 257,\ \delta = 0.7$	9138	9-33	1.7070
RAP6	$\sigma = 257,\ \delta = 1$	24920	9-40	1.8494
CC biclusters				
CC1	$\delta = 0.3$	100	187-3249	∞
CC2	$\delta = 0.5$	100	273-2940	∞
SAMBA biclusters				
SAMBA1	# probes = 10	10	234-1314	∞
SAMBA2	# probes = 100	349	120-1450	∞

Table 1: Statistics of biclusters generated at various parameter settings from the GI data set

4.3 Statistical significance of RCBs

One of the important steps in analyzing real-life data, such as the GI data in our study, is to assess the validity and significance of the entities being mined from them. A common method for doing this is to randomize the data and mine the same type of entities from this randomized data set. A meaningful analysis should find significantly more of these entities from the real data as compared to the randomized version. We performed such an analysis for RCB mining from GI data. We found 12099 RCBs in the RCB4 set using $\delta = 0.5$ and $\gamma = 40$ from the GI data set. Also, we generated 30 randomized versions of this data set by randomly permuting the entries in each row, and derived RCBs from them using the same parameter settings as RCB4. Two observations can be made from these results. First, very few RCBs (348

[1]http://vk.cs.umn.edu/gaurav/rap/

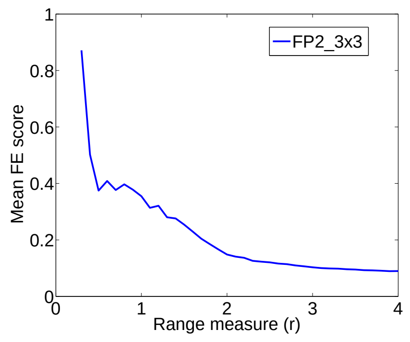

Figure 2: Relationship between Range (r) and Functional Relatedness (FE).

on average) are found from the randomized data sets. Second, the sizes of these RCBs are substantially smaller than the sizes of those in the RCB4 set, with most of the RCBs in the RandRCB4 (over 90%) being 3×3 blocks. The results of this analysis indicate that the products of our RCB mining approach are indeed statistically significant. The biological significance of some of these blocks is discussed in the next section.

4.4 Functional Evaluation

Since the groups of genes constituting a submatrix with coherent values are expected to be functionally coherent, we evaluate this using a measure of functional relatedness derived from sources of information about gene function that are independent of GI data. In particular, we use Functional-coExpression (FE) that is derived from 40 different micro-array data sets [12]. Here the probability for two genes to be co-annotated to the same Gene Ontology Biological Process (GO BP) function is computed on their levels of co-expression in these datasets. We refer interested readers to the corresponding paper for details on this measure, but stress that the basic purpose is to quantify the degree of functional relatedness of two genes.

Genes constituting both the dimensions of a submatrix are said to be functionally related, if each gene-pair which is a combination of one gene from each group has high functional-coExpression score. So, for any given submatrix, we compute the functional relatedness as the mean of the FE score for each interacting gene pair covered by the submatrix. Although it is possible to evaluate the relationship between range and the FE score on RCB patterns, the average range of the binary patterns is higher (as shown in Table 4.2) and so we use the binary patterns to evaluate this relationship. We evaluated the relationship between the functional relatedness and the range measure r, by enumerating all possible 3×3 size blocks for randomly chosen 10,000 patterns in FP2, which we refer to as FP2_3×3.

The FE score and the range are computed for each such 3×3 size block enumerated. The median of the FE scores for corresponding range values are presented in Figure 2. It can be seen that the blocks with a small range value, have high FE score and the blocks with large range value has low mean FE score. This indicates that the groups of genes representing the coherent submatrices are more functionally related than the groups of genes representing the less coherent submatrices.

4.5 Comparison of our RCB finding algorithm with post-processing of FP and RAP

It is also possible to enumerate all possible submatrices from binary patterns and RAP patterns and select the submatrices that satisfy a given range threshold. To demonstrate the effectiveness of RCB, we enumerated all possible submatrices that satisfy the range constraints for FP1, FP2, RAP1, RAP2 and RAP3. As the size of the CC and SAMBA patterns are typically large relative to the FP and RAP patterns, enumerating all possible submatrices is infeasible. So, we restrict our analysis to FP and RAP patterns. For FP1, the range threshold $\delta = 0.3$ and $\delta = 0.5$ are used to compare them with RCB1 and RCB2 respectively. For FP2, the range threshold $\delta = 0.3$, $\delta = 0.5$, $\delta = 0.7$ and $\delta = 1$ are used to compare them with RCB3, RCB4, RCB5 and RCB6 respectively. For RAP patterns the δ that was used in Table 1 was chosen. For each set of patterns, the number of genes covered in both the dimensions, total number of interactions covered i.e. the area in the data matrix that the discovered patterns cover, time taken are tabulated in Table 2.

Title	Range (δ)	# Genes covered	# Interactions covered	Time taken (in hours)
RCB biclusters				
RCB1	0.3	(408, 2437)	26664	1.62
RCB2	0.5	(484, 3391)	54842	3.2
RCB3	0.3	(216, 765)	4959	0.29
RCB4	0.5	(327, 1594)	16550	0.41
RCB5	0.7	(371, 1986)	22516	0.8
RCB6	1	(415, 2234)	26054	7.12
Binary Patterns				
FP1a	0.3	(293, 641)	6169	0.2
FP1b	0.5	(433, 1263)	22947	0.32
FP2a	0.3	(170, 421)	2642	0.06
FP2b	0.5	(286, 981)	10858	0.07
FP2c	0.7	(340, 1262)	16394	0.1
FP2d	1	(384, 1447)	20034	0.3
RAP biclusters (on query genes)				
RAP1	0.5	(53, 303)	1467	0.04
RAP2	0.7	(89, 756)	4959	0.06
RAP3	1	(111 ,1123)	8607	0.23
RAP biclusters (on array genes)				
RAP4	0.5	(156, 277)	2404	0.13
RAP5	0.7	(212, 502)	5987	0.28
RAP6	1	(280, 658)	9648	0.71

Table 2: Comparison of RCB with FP and RAP (with post processing) at various range thresholds.

RCBs cover more number of genes in both dimensions than FP and RAP patterns. Specifically, comparing the coverage of the sets RCB1 and FP1a, RCB1 covers approximately twice as many genes covered by FP1a in the query dimension and four times as many genes in the array dimension. They also cover four times as many interactions covered by the FP1a. This difference is relatively less at high sparsification thresholds γ, due to the sparse nature of the resulting binary matrix. The coverage of the genes and interactions is much less for the RAP patterns. This is due to the fundamental difference between the RCB approach and the general frequent pattern based approach. The RCB approach builds blocks in a bottom up fashion starting with a 1×1 block and gradually increasing its size in either dimensions while using range measure to control the complexity of the search space. On the other hand, FP based approaches start with single item that can have some support, which monotonically decreases as the size of itemset increases. The use of high support thresholds needed to contain the complexity resulting from the high density of the data prevents FP based approaches from discovering small patterns that RCB can capture. For example, FP1 patterns are generated using $\sigma = 6$ which means each pattern generated should have atleast 6 genes on the query dimension. On the other hand, the coherent blocks of smaller sizes (of the order 3×3) exist in a large number compared to the bigger blocks in the data set (as shown in Table 4.2). So, these patterns cannot be discovered using the traditional frequent pattern based approaches especially at low sparsification thresholds, whereas RCB can discover all such patterns for a given range threshold δ.

On the other hand, from Table 2 the time taken for RCB generally appears to be larger than that of FP patterns, but note that the RCB patterns usually cover many more number of interactions than the FP patterns. Considering the set of biclusters in RCB6 and FP2d, RCB6 appears to require more time, but FP2d covers less number of interactions that of RCB6 because of a support threshold of 4, which causes it to miss many patterns of size 3×3. Note that the lowest possible support is being used to generate the FP patterns without 'running out of memory'. Similarly, the time taken for discovering RCB2 is more than that of FP1b. However, the number of interactions covered by RCB2 is more than twice as many as FP1b, also due to the use of high support to contain the complexity resulting from the density of the matrix at sparsification level $\gamma = 30$. This indicates that the RCB is an efficient and systematic approach to discover all the submatrices with range less than a given threshold than the other approaches.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we presented a novel association analysis framework for mining (nearly) constant value submatrices from real valued genetic interaction datasets. We evaluated the proposed RCB discovery approach and compared its performance with other approaches, namely binary frequent patterns (FP), RAP, CC and SAMBA. Our results show that the gene modules representing the biclusters with similar values are more functionally related than the gene modules representing biclusters with diverse values. Furthermore, our approach can exhaustively find all the biclusters with range r less than a given threshold. This is not possible with other approaches, even when they are coupled with an exhaustive post-processing phase to enumerate submatrices with range within a given δ. Finally, we have shown that the RCBs discovered are statistically significant and are also biologically meaningful. This work can benefit from further research in many directions. The process of discovering RCBs can be made faster using specialized data structures and algorithms, such as hash trees. Our approach like other association analysis based approaches, provides a large number of patterns, many of which may be slight variation of the other patterns. Summarization techniques such as those in [22] will be helpful for the effective utilization of RCB patterns in practical settings.

6. ACKNOWLEDGEMENTS

This work was supported by NSF grants #IIS0916439, #CRI-0551551, a University of Minnesota Rochester Biomedical Informatics and Computational Biology Program Traineeship Award. Access to computing facilities was provided by the Minnesota Supercomputing Institute.

7. REFERENCES

[1] R. Agrawal et al. Mining association rules between sets of items in large databases. In *Proc. SIGMOD*, pages 207–216, 1993.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. VLDB*, pages 487–499, 1994.

[3] L. Avery and S. Wasserman. Ordering gene function: The interpretation of epistasis in regulatory hierarchies. *Trends in genetics*, 8(9):312, 1992.

[4] S. Barkow et al. BicAT: a biclustering analysis toolbox. *Bioinformatics*, 22(10):1282–1283, 2006.

[5] A. Ben-Dor et al. Discovering Local Structure in Gene Expression Data: The Order-Preserving Submatrix Problem. *JCB*, 10(3-4):373–384, 2003.

[6] S. Bergmann et al. Iterative signature algorithm for the analysis of large-scale gene expression data. *Physical Review*, 67(3):031902, 2003.

[7] C. Borgelt. Efficient implementations of apriori and eclat. In *FIMI*, 2003.

[8] A. Ceglar and J. F. Roddick. Association mining. *ACM Comput. Surv.*, 38(2):5, 2006.

[9] Y. Cheng and G. Church. Biclustering of Expression Data. In *Proc. ISMB Conference*, pages 93–103, 2000.

[10] I. Dhillon et al. Information-theoretic co-clustering. In *Proc. SIGKDD*, pages 89–98, 2003.

[11] J. Han et al. Frequent pattern mining: current status and future directions. *DMKD*, 15:55–86, 2007.

[12] C. Huttenhower, M. Hibbs, C. Myers, and O. G. Troyanskaya. A scalable method for integration and functional analysis of multiple microarray datasets. *Bioinformatics*, 22(23):2890–2897, 2006.

[13] R. Kelley and T. Ideker. Systematic interpretation of genetic interactions using protein networks. *Nature biotechnology*, 23(5):561–566, 2005.

[14] X. Ma et al. Mapping genetically compensatory pathways from synthetic lethal interactions in yeast. *PLoS ONE*, 3(4):e1922, 2008.

[15] S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM TCBB*, 1(1):24–45, 2004.

[16] G. Pandey et al. Association Analysis Approach to Biclustering. In *Proc. SIGKDD*, 2009.

[17] R. Shamir et al. EXPANDER – an integrative program suite for microarray data analysis. *BMC bioinformatics*, 6(1):232, 2005.

[18] R. St Onge et al. Systematic pathway analysis using high-resolution fitness profiling of combinatorial gene deletions. *Nature Genetics*, 39(2):199–206, 2007.

[19] A. Tanay et al. Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, 18(90001):S136–S144, 2002.

[20] A. Tong et al. Global mapping of the yeast genetic interaction network. *Science*, 303(5659):808–813, 2004.

[21] I. Ulitsky et al. From E-MAPs to module maps: dissecting quantitative genetic interactions using physical interactions. *Molecular Systems Biology*, 4(1), 2008.

[22] C. Wang and S. Parthasarathy. Summarizing itemset patterns using probabilistic models. In *KDD*, page 735, 2006.