Graph Clustering to Identify Allele-Specific Interactions in Psoriasis GWAS

Sayali Patil
sayali.patil@wustl.edu
Department of Computer Science and Engineering
Washington University in St. Louis, St. Louis, USA

ABSTRACT

Psoriasis is a complex skin disease with known genetic risk factors. Recent literature has explored the use of network-representations of allele-interactions to identify sets of alleles with high interaction and correlation to psoriasis. In this study, we evaluate the use of alternate graph clustering techniques on genome-wide allele-specific networks to overcome limitations in prior work; present an extensive evaluation of the methods and the allele-clusters identified by them. We also present runtime analysis of clustering techniques, discuss their scalability and propose future directions.

KEYWORDS

psoriasis, allele-interactions, graph-clustering

1 INTRODUCTION

Psoriasis is an incurable, complex skin disease that occurs in about 2-3% of the world's population. Symptoms of psoriasis include red, patchy scales on one's skin. The disease is known to have several environmental triggers, and medical research in potential genetic risk factors has gained steam over the years. Genome-wide association studies (GWAS) have identified certain genetic risk factors for psoriasis. Psoriasis has a strong genetic predisposition with an 80% estimated heritability [2, 4].

Prior work in identifying interacting groups of psoriasis-correlated alleles using genome-wide allele-specific network representations [2] provides evidence for the use of graph-clustering techniques to discover such interactions. The method uses a Custom Correlation Coefficient (CCC), a quantitative metric to measure allele-specific interactions in terms of co-occurrence across individuals. It then constructs a network with these interactions forming edge-weights over a graph with alleles as nodes, and uses Breadth First Search (BFS) to uncover tightly knit groups of alleles in this graph.

However, this method has several drawbacks. First, to be able to use BFS on a network to uncover clusters, the network must not be fully connected. To achieve sparsity, the method imposes an arbitrary threshold on the CCC values to retain as edges. Even though the threshold is shown to be significantly higher than would occur by random chance, there is no empirical evidence of this threshold being optimal. Secondly, BFS is limited to finding connected components, that is, a cluster may not be part of a larger set of connected nodes. This is extremely restrictive as clusters being a component are a consequence of the arbitrarily chosen threshold.

To this end, we propose and evaluate the use of 5 different graph-clustering methods to uncover underlying allele-interactions correlated with psoriasis. We present methodological advances in the handling of genomic data in the form of bit-arrays for memory compactness, allowing for better runtime efficiency. We compare results to the baseline qualitatively and quantitatively, providing further insight into the use of such techniques.

2 RELATED WORK

In [2], the authors propose BlocBuster, a method used to identify sets of allele-interactions which are highly correlated with psoriasis, using a network-representation of said allele-interactions. To accomplish this, BlocBuster uses breadth-first search (BFS) on a weighted, undirected graph where each edge represents the level of interaction between two alleles, quantified with a custom correlation coefficient (CCC). Having identified components in the graph, the method proceeds to identify carriers of the set of alleles across case and control populations, testing for occurrence more significant than those that occur by random chance. We use this paper as a baseline in our work, comparing the efficacy of various clustering methods to BFS in identifying close-knit allele groups.

The custom correlation coefficient (CCC) metric was proposed in [3], as an improvement over commonly used methods like Pearson correlation coefficient (PCC) and r^2 coefficient of determination. Given an individuals genome represented as a set of bi-allelic SNPs, PCC and r^2 only account for correlations between SNPs as a whole, while CCC returns allele-specific interactions.

	P1	**P2**	**P2**	**P4**	**P5**	**P6**	**P7**	**P8**	**P9**	**P10**
SNP1	Aa	AA	AA	AA	AA	AA	AA	AA	AA	AA
SNP2	bb	Bb	bb	bb	bb	bb	bb	bb	bb	bb

Table 1: Genotypes of ten individuals for a pair of SNPs

This method is designed to work with bi-allelic SNPs, where each SNP may contain either of the 4 nucleic acids - A, C, G, T. Between a pair of SNPs CCC first looks for all the possible combinations of alleles. Let 'a' and 'A' be the two alleles of SNP1 and 'b' and 'B' be the two alleles of SNP2. This forms four possible allele pairs as - ab, aB, Ab, and AB. Over the entire population, we count the frequencies of co-occurrences for each such unique allele pair in the dataset, along with the frequency count of each individual allele per SNP. As an example, refer to 1, where the pair 'Ab' occurs 36 out of 40 times, and the frequency of allele 'A' in SNP1 is 19 out of 20.

To quantitatively capture allele-interactions, CCC first considers the average weight of a relationship between allele 'i' and allele 'j'. In the example given above, the relationship 'Ab' occurs with an average weight equal to 0.90. Between a pair of SNPs, the sum of average weights of four possible allele combinations thus always equals 1; $R_{ab} + R_{aB} + R_{Ab} + R_{AB} = 1$.

CCC then uses the following frequency factor to account for the frequency of rare alleles:

$$F_i = 1 - \frac{f_i}{q}$$

where f_i is a frequency of allele i and q is a tuning parameter that is set to 1.5. The value of q is obtained empirically in [3]. To compute the CCC value of a relationship between two alleles, R_{ij} is multiplied with the frequency factors of corresponding alleles i and j. The resulting value is further multiplied by $\frac{9}{2}$ to re-scale the correlation values between 0 and 1. Together, the CCC value between alleles i and j is given by:

$$CCC_{ij} = \frac{9}{2} * R_{ij} * F_i * F_j$$

For the relationship 'Ab' in 1, the frequency of allele 'A' in SNP1 is 0.95 and the frequency of allele 'b' in SNP2 is 0.95. Based on these frequencies the frequency factors and the average weight of the relation 'Ab', we get a CCC value of 0.5447 for this particular allele-allele interaction. On the other hand, when we run PCC and r^2 metrics on this sample dataset, both the measures return a low correlation value of 0.1 and 0.0 respectively.

3 GRAPH CLUSTERING - PRELIMINARIES

Next, we turn our attention to the various graph-clustering methods considered in this study, as alternatives to the BFS step in BlocBuster.

3.1 Markov Clustering

The Markov Clustering (MCL) algorithm was proposed in [14], and is based on the intuition that if a random walk is started in or near a dense clique in a graph, it is more likely to get trapped in the same cluster than traveling between two such dense areas connected by a low weight edge. For a weighted undirected graph, $G = (V, E)$, and an adjacency matrix A of G with the inclusion of self-loops for all the nodes in the graph, the algorithm starts by constructing an initial flow matrix (here flow refers to probabilistic information flow between nodes) named the Markov matrix M, alternatively called the transition matrix. Every entry in M_{ij} in the matrix is defined as:

$$M_{ij} = \frac{A_{ij}}{\sum_{k=1}^{n} A_{kj}}$$

Every such entry in M represents the probability of a transition from node i to node j. MCL is an iterative method where it updates the transition matrix M until a stable state is achieved (convergence). Every iteration involves the following three steps:

- Matrix Expansion: $M_t = M_{t-1} * M_{t-1}$
- Matrix inflation: $M_{t(ij)} = \frac{M_{t(ij)}^r}{\sum_{k=1}^{n} M_{t(kj)}^r}$, where r is an inflation parameter > 1.
- Matrix Pruning: $M_{t(ij)} = 0$, if $M_{t(ij)} <$ threshold. Every column is normalized after pruning to sum to 1.

A random walk on G corresponds to taking successive powers of the transition matrix M (exapansion) per iteration, which results in longer random walks. Matrix inflation helps with increasing the higher probability transitions and decreases the ones with lower probabilities. When MCL achieves a steady state, there is at least one non-zero element found in every column of M. All nodes in G which have a non-zero element in the same row in their corresponding columns in M are assigned the same cluster label. Multiple non-zero elements in the column result in a random tie-break. MCL was successfully applied to protein family detection in [5].

The alternating expansion and inflation steps adjust the transition probabilities in different regions of the graph. Intuitively, this process favors regions in the graph containing dense sets of connected nodes, while making transitions less likely across edges with low weight. We expect these low weight edges to be edges between different clusters - in our case, this translates to sets of well-interacting alleles separated from each other by low interactivity. The tuning of graph sparsity further helps identify tight clusters for our datasets, as low-weight edges are initially dropped, resulting in faster convergence.

3.2 Affinity Propagation

Affinity Propagation (AP) [6] clusters data points by passing messages between them. For a graph $G = (V, E)$, nodes in the graph act as data points. These data points can either be potential cluster centers, called exemplars or other non-exemplar data points. When the algorithm starts, every node in the graph receives messages from every other node in the graph. These messages evaluate the suitability of each node to serve as an exemplar for other nodes in the graph. AP is also an iterative procedure in which messages from the previous iteration are used in the next. The input to the algorithm is a similarity or affinity matrix, S. Using the values in S, the algorithm iteratively builds a responsibity matrix R and an availability matrix A. Before the start of first iteration, every element in the availability matrix is set to zero.

For a data point i and an exemplar k, an entry R_{ik} in the responsibity matrix is used to infer the willingness of data point i to join a candidate exemplar k. Responsibility is given as:

$$R_{ik} = S_{ik} - \max_{k' \neq k} \{A_{ik'} + S_{ik'}\}$$

Availability A_{ik} is used by a candidate exemplar to inform other data points of its suitability to serve as an exemplar. While calculating the availability matrix two different formulae are used to update the availability of on-diagonal and off-diagonal elements. The availability of elements on the diagonal is called as self-availability where each candidate exemplar evaluates itself as an exemplar.

Availability A_{ik} is given as:

$$A_{ik} = \min\{0, R_{kk} + \sum \max_{i' \neq i} \{0, R_{i'k}\}\}$$

Self-availability is given by:

$$A_{kk} = \sum \max_{i' \neq k} \{0, R_{i'k}\}$$

Once the algorithm converges, we get a criterion matrix C, where each entry C_{ik} is the sum of the responsibility and availability matrix entires at that location. It is given as:

$C_{ik} = R_{ik} + A_{ik}$

The cluster assignment step in the algorithm is similar to that of MCL. In a row of C, the element with the highest value is assigned as an exemplar and rows sharing the same exemplar form

one cluster. Clusters found by AP can qualitatively differ from other methods; especially MCL and commDet. Consider an example, as demonstrated in [16], of two clusters separated by a high-weight edge. While methods that account for network-structure would divide these into two sets of nodes, AP may assign them to the same cluster. This property may be undesirable in applications like community detection or partitioning problems, but when identifying allele-interactions, this may in fact help consolidate clusters of alleles that interact with each other, avoiding over-partitioning.

3.3 Community Detection by Information Flow Simulation

Next, we consider a heuristic approch to graph clustering as proposed in [15]. This approach leverages parallel message-passing to reduce time-complexity. Community detection by flow simulation (henceforth referred to as commDet) differs from conventional clustering methods, in that not all the nodes in a graph are necessarily assigned to a cluster. Similar to Markov clustering, this method also relies on the notion of random walks being captured inside dense communities. The algorithm starts by looking for nodes that have high weighted as well as unweighted degrees - essentially nodes that have a large number of *important* neighbors. It assumes that these nodes, called alpha-nodes, are highly likely to be cluster centers or well-connected within a cluster. For clustering, this algorithm assumes that the graph contains dense local structures, and roughly follows a power-law degree distribution.

To find alpha nodes, the method sorts two copies of the vertex set; one by weighted degree and one by unweighted degree. It then retains the top $k\%$ of nodes in each list (k here is a tuneable parameter for this approach), and takes the intersection of these subsets. Then from each alpha-node u, the algorithm propagates the node's label (which in turn, is the cluster label) to all neighbors, v probabilistically at each step, with the following likelihood:

$$p(u, v) = \left(\frac{w_{uv}}{\delta^*(u)} \right)^{\beta}$$

where w_{uv} is the weight of the edge between nodes u and v, and $\delta^*(u)$ is the node's total weighted degree. $\beta \in (0, 1]$ is a scaling parameter, similar to MCL's inflation parameter, skewing probabilities closer to 0 or 1. While this method resembles MCL by following intuition about random walks, there are a few key differences. Multiple random walks are started from each alpha-node, and walks do not cross nodes that have already been visited by the same, or a different random walk. Therefore there is a constant out-bound propagation of information from each alpha-node, and the clusters are covered in a probabilistic breadth-first manner. Finally, there is no pruning step after propagation at each iteration. Qualitatively, the clusters identified by this method can be different from those identified by MCL. While MCL identifies clusters that necessarily have high interactions among a majority of the cluster's nodes, commDet can also identify clusters that are predominantly star-like structures with high central-node degree.

3.4 Spectral Clustering

Finally, we consider Spectral Clustering [9], a commonly used technique for graph partitioning. Spectral clustering maps data points to a lower-dimensional space using the k smallest eigenvalues and eigenvectors of the graph Laplacian, constructed as follows. Given adjacency matrix, A, and a degree matrix D with the sum of incident edge weights for each node along the diagonal, the normalized graph Laplacian is given as:

$$L_N = I - D^{-1}A$$

where I is the identity matrix. Having obtained the low-dimensional representation matrix, V, the cluster labels are assigned by running K-Means clustering on the rows of V. Alternatively, we use K-Means with discretization in the final step to reduce sensitivity to random noise from the initializations of cluster centers.

The lower-dimensional projection step in spectral clustering encodes information about data-point similarity, while making them highly-separable in this projection. Further, doing this also allows us to find clusters with non-convex boundaries. This is especially relevant to our data, where the adjacency matrix provides a high-dimensional feature representation for each node, where a large number of entries may be of low magnitude. Projecting these relationships into lower dimensions easily lets us capture groups of similar nodes. Indeed, our results section shows that spectral clustering is capable of finding significantly bigger clusters as compared to the others considered in the study.

4 MATERIALS AND METHODS

4.1 Data and Preprocessing

The data for this study were obtained from the Genetic Association Information Network (GAIN) [7], consisting of sampled genotypes. We obtained two datasets - General Research Use (GRU) and Autoimmune Disease Only (ADO) respectively. These original datasets contain 430,020 Single Nucleotide Polymorphisms (SNPs) per individual. Further, both datasets are divided into case and control populations. The GRU dataset contains 929 cases and 681 controls, whereas ADO contains 439 cases and 728 controls. The dataset can be accessed at http://view.ncbi.nlm.nih.gov/dbgap-controlled through dbGaP accession number phs000019.v1.p1.c1.

The dataset consists of rows that represent SNPs and columns that represent individuals. For this study, we only consider bi-allelic SNPs (two alleles for a given SNP). Each entry (i.e. each SNP for a given individual) consists of a nucleic acid pair (A, T, G or C). We also have the major and minor allele for each SNP in the dataset, along with a *dbSNP_ss ID*, the submitted SNP variant based on asserted location or flanking sequences.

4.1.1 Computational Limitations and Subsampling. To construct a graph of allele-interactions requires counting the occurrence of each unique allele-pair in the dataset. For n SNPs, this amounts to $\binom{2n}{2}$ combinations. Fig. 1 visually represents this combinatorial growth rate.

Because of limited computing hardware at our disposal, it was impossible to load a graph with such high edge-count into memory for the entire dataset. Therefore in this study, we consider a subset of 5000 SNPs, which leads to a graph in the worst case of about 50 million edges.

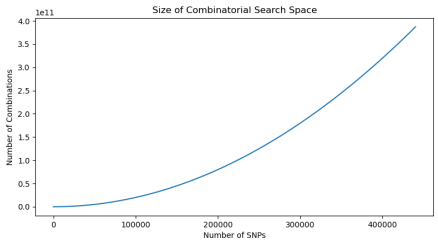

Figure 1: Growth in number of possible combinations of alleles, with increasing number of SNPs

4.1.2 BitArray Representation. To further optimize memory consumption, we converted the data into an array-of-bits representation as follows. For each SNP, let 0 represent the major allele and 1 represent the minor allele. Then for an individual, the two allele entries would each be 0 or 1, depending on which variation of the allele they carry, therefore leading to a pair-of-bits per SNP format. This enables time-efficient bitwise operations as compared to string manipulation.

4.2 Computing CCC values

We first combine both cases and controls for the ADO and GRU datasets, following the baseline paper. Values of a given SNP for all individuals is represented as a vector of bit-pairs. For each SNP, we divided the CCC computations with remaining SNPs into multiple processing threads. In each thread, at each step, we deal with four bitarrays - one for each allele for each of the 2 SNPs considered. Using dot products and the binary not operator allowed us to quickly calculate the total number of occurrences of all possible allele-combinations in the 2 SNPs under consideration. This led to a running time of about 3-4 hours for each dataset.

We follow the BlocBuster paper in dealing with missing data; missing data is not imputed prior to computing CCC values, to prevent imputation error bias. Further, individuals with missing genotypes in either SNP in a pair are omitted from the CCC calculations. We as a consequence, keep a count of individuals dropped in this process to adjust the population size when normalizing.

4.2.1 Assessing Type I errors. To check if significant CCC values could arise by random chance in our limited dataset, we conducted permutation trials, following the steps taken by the baseline paper. Here, for each SNP, we permuted the genotypes across individuals, while maintaining global frequencies and alleles. This process only destroys inherent correlations between pairs of SNPs. CCC values were computed for 5 permutations of the data, and we observed that the highest CCC value that occurs by random chance is significantly lower (0.65) than the original data (0.78).

4.3 Graph Construction

After CCC computations, we begin by constructing an allele-interaction graph where each allele is represented by a node and each SNP contributes 2 nodes to the graph - one for each allele. An edge exists between two nodes if there is a non-zero correlation between a pair of alleles. Fig. 2 shows one such instance where the graph is fully connected.

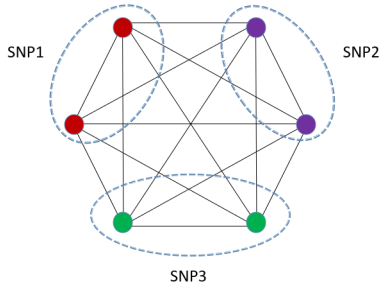

Figure 2: Fully connected network with 3 SNPs and 6 alleles.

In the BlocBuster paper, the authors use an arbitrary threshold on the CCC values (although it is shown to be higher than random) to reduce the number of edges in this graph. They also then only retain the top n edges in the graph, where n is the number of SNPs. On the other hand, we vary the network density empirically for each method considered, except when running BFS where we followed the baseline for replication.

Each graph clustering method considered in this study is designed for networks with certain degree-distributions. Affinity propagation and spectral clustering make no assumptions about the internal structure of the graph, and as such can be used directly on the fully connected network without imposing any threshold on edge-weights. MCL and commDet on the other hand are suited to graphs that follow a power-law degree-distribution. Fig 3 shows the change in degree distributions as the top k edges are retained, with increasing k. For these methods, while we ran experiments with the entire range of number of edges, only those levels of sparsity where the distribution was still power-law like gave us meaningful results.

4.4 Clustering - Implementation

For MCL and commDet, we use the respective authors' original implementation, available freely. For AP, BFS, and Spectral Clustering, we use the Python 3 implementations in the Scikit-Learn [10] library. Here we note the values of the various hyperparameters used in this study for replication, as well as the sparsities of the network considered for each method. Where sparsities were varied (MCL and commDet), the number of edges retained was chosen from n=[5k, 7.5k, 12.5k, 25k, 50k, 100k, 250k, 500k, 1M, 3.125M, 6.25M, 12.5M, 25M, 50M (full graph)]. For BFS, top 5000 edges are retained. For AP and Spectral clustering, the entire graph is used. All methods were run on both ADO and GRU datasets.

- **MCL** - Inflation parameter from [1.4, 2, 3, 4, 5, 6].
- **AP** - Damping factor from [0.5, 0.6, 0.7, 0.8, 0.9]. Max iterations = 1000. Convergence after 15 idle iterations. Self-affinities set to 1.
- **commDet** - k for alpha-nodes from [0.5, 0.6, 0.7, 0.8, 0.9]. Scaling factor = 0.25. Convergence after 15 idle iterations.
- **Spectral Clustering** - Number of clusters between 100 to 1800 in increments of 100. KMeans and Discretization iterations set to 100.

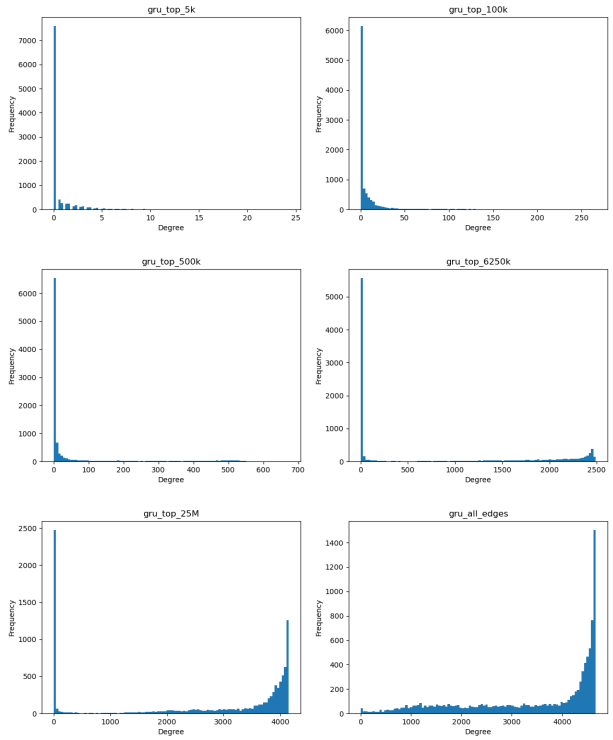

Figure 3: Degree Distributions when number of retained edges is varied

4.5 Interpreting Clusters

4.5.1 Carriers and Odds Ratios. Next, we seek to identify individuals who carry an entire cluster of alleles as identified by our methods in their genomes. To achieve this, we sample the genome bit-pair arrays at the positions of interest in the cluster, split the two alleles and use a binary equation to check if the individual's genotypes match the cluster at all positions. This, combined with the bit-array representations allows for vectorized operations. We check for the number of carriers in cases and controls for both GRU and ADO datasets for each given cluster.

Having identified carriers, we compute the odds ratio (OR) for each cluster as follows:

$$OR = \frac{p(1-q)}{q(1-p)}$$

where p and q are the frequencies of a cluster for cases and controls respectively. The odds ratio is a metric used to evaluate significance of clusters found. Clusters with higher odds ratios are thought to be of significant phenotypic associations. We retained the top 15 clusters by OR for each method, per choice of hyperparameter and network sparsity. These top OR clusters in the GRU dataset were compared to clusters in the ADO datasets - both in terms of complete cluster match, as well as significant non-null intersections, as clusters may only partially overlap. We checked for these intersections a) when using the same method on both datasets, as well as b) when using different methods - one on each dataset.

4.5.2 Gene Lookup. The strongest known genetic risk factor for psoriasis is located on Chromosome 6 of the human genome; this region is known as the Major Histocompatibility Complex (MHC) region [1, 8, 12, 13]. The genetic factors of interest for psoriasis mainly include HLA-C, PSORSIC1, PSORSIC2, PSORSIC3 and CDSN. There are several other genes and pseudogenes present in the region of 6p21.3, which are also correlated with psoriasis.

To evaluate the clusters that we found in both GRU and ADO datasets, we implemented automated gene-lookup with web-scraping techniques using the Python BeautifulSoup library [11]. Each SNP in our data has an associated *dbSNP_ss ID*, which in turn corresponds to an *rsID* - the reference SNP ID. We first lookup the *dbSNP_ss ID* on https://www.ncbi.nlm.nih.gov/snp, and then visit the page corresponding to the relevant *rsID*. From this page, we then extract the gene consequence for each SNP. If a cluster has a large number of SNPs that have gene consequences relevant to psoriasis, we posit that these allele-interactions are significantly related to the disease.

5 RESULTS

5.1 Evaluating Clusters

First, we present the results from when we find entire clusters matching across GRU and ADO for each of our methods, along with their odds ratios and the count of SNPs relevant to psoriasis in that cluster in Table 2. We observe that BFS only identifies 4 such clusters, of which 2 are doubletons. Of the remaining two, one is completely correlated with psoriasis, and the other is completely uncorrelated. AP gives a single such cluster with relatively high OR, and number of correlated SNPs. MCL can identify clusters of varying sizes with high correlation to psoriasis. The commDet method identifies small clusters, but they are almost always entirely comprised of correlated SNPs. Spectral clustering with K-Means outperforms Spectral clutering with discretization in terms of ORs, but both find relatively small clusters.

Next, we present the results when clusters partially overlap between the GRU and ADO datasets in Table 3. Here, we observe that BFS, AP, MCL and commDet all identify somewhat large clusters, where a majority of SNPs are seen to be correlated with psoriasis. These clusters also have fairly high odds ratios. Spectral clustering with K-Means identifies smaller clusters with odds ratios similar to other methods, but spectral clustering with discretization identifies clusters with low odds ratios. However, two of the clusters identified by the latter consist of 84 and 21 SNPs respectively, with 51 and 15 correlated SNPs each.

Finally, we present three clusters that were identified in their entirety by three different methods in Table 4. These clusters were identified by BFS, MCL and commDet, and two out of these are comprised of SNPs correlated with psoriasis.

5.2 Runtime Analysis

In this section we present a comparison of runtimes for the various clustering methods used in this study. First, we present runtimes for each method on the fully-connected graph in Fig. **??** - even though

Method	GRU OR	ADO OR	No. of SNPs	No. of correlated SNPs
BFS	1.99	2.48	2	1
BFS	1.66	2.05	4	0
BFS	1.65	1.66	2	1
BFS	1.61	2.25	5	5
AP	2.09	2.13	11	6
MCL	2.19	1.65	4	2
MCL	2.04	1.76	3	3
MCL	2.01	1.38	3	3
MCL	1.96	1.63	30	21
MCL	1.79	1.64	22	18
commDet	2.05	2.31	5	4
commDet	1.76	1.63	3	3
commDet	1.67	2.26	4	4
commDet	1.60	2.25	5	5
commDet	1.46	2.11	4	4
Spectral KM	2.33	2.78	3	3
Spectral KM	2.09	2.13	4	4
Spectral KM	1.98	2.31	3	1
Spectral KM	1.48	1.33	7	4
Spectral KM	1.08	1.14	3	1
Spectral DS	1.47	1.28	9	5
Spectral DS	1.32	1.17	4	1
Spectral DS	1.23	1.19	2	2
Spectral DS	1.22	1.02	4	1
Spectral DS	1.14	0.96	3	0

Table 2: Evaluation of complete clusters found in GRU and ADO using different methods.

Method	GRU OR	ADO OR	No. of SNPs	No. of correlated SNPs
BFS	2.25	2.44	24	18
BFS	1.96	1.88	14	9
BFS	1.79	1.78	8	6
BFS	1.73	1.71	18	3
BFS	1.66	1.67	12	4
AP	2.43	2.84	29	20
AP	2.30	2.75	29	20
AP	1.69	1.40	26	14
MCL	2.32	3.12	11	10
MCL	2.26	2.91	23	17
MCL	1.88	2.53	10	8
MCL	1.73	1.90	8	7
MCL	1.69	2.26	25	19
commDet	3.28	2.33	4	3
commDet	3.28	1.78	6	5
commDet	3.28	1.43	7	5
commDet	3.28	1.88	5	5
commDet	2.78	1.92	14	13
Spectral KM	3.16	3.49	3	2
Spectral KM	1.97	2.15	6	4
Spectral KM	1.92	2.20	8	6
Spectral KM	1.76	1.46	7	6
Spectral KM	1.19	1.15	3	3
Spectral DS	1.17	1.01	4	3
Spectral DS	0.99	0.94	21	15
Spectral DS	0.80	1.09	7	5
Spectral DS	0.80	0.79	84	51
Spectral DS	0.74	0.60	6	5

Table 3: Evaluation of overlapping clusters found in GRU and ADO using different methods.

the results from such a run may not be meaningful, this presents a worst case running time for a given number of SNPs. Spectral clustering with K-Means takes the longest, while commDet and AP have significantly lower runtimes. Here, the damping factor for AP was set to 0.8, MCL's inflation parameter was set to 4, commDet's k value was set to 0.7 and number of clusters for the 2 spectral clustering methods was set to 800.

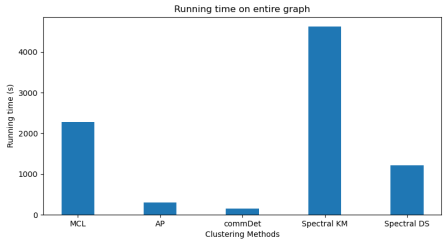

Figure 4: Running times for various clustering methods on fully connected graph of 50 million edges.

For methods where sparse graphs are used (MCL and commDet), we present results for how runtime scales with the increase in the number of edges, in Fig. 5. For this analysis, MCL's inflation parameter was set to 4 and commDet's k value was set to 0.7. With an increase in graph size, we observe that commDet scales much better in terms of runtime than MCL.

Method	GRU OR	ADO OR	No. of SNPs	No. of correlated SNPs
BFS, MCL, commDet	1.66	2.05	4	0
BFS, MCL, commDet	1.62	1.89	3	3
BFS, MCL, commDet	1.60	2.25	5	5

Table 4: Evaluation of clusters common to different methods.

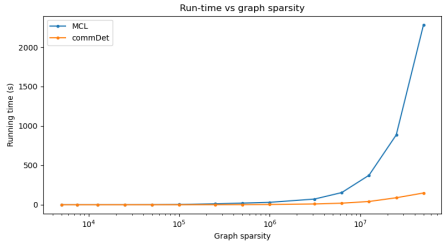

Figure 5: Running times for MCL and commDet on networks of varying sparsity.

6 DISCUSSION

In this project, we explored several graph clustering techniques on a subset of 5000 SNPs as a proof-of-concept for the use of such techniques in identifying allele-interactions correlated with psoriasis. Our results show that these methods may be useful in a) avoiding arbitrary thresholding on CCC values during graph construction, b) indentifying a wider range of cluster sizes, c) identifying clusters that are not necessarily connected components, and d) identifying psoriasis-correlated allele-interactions present within a wider range of genes. We acknowledge the limitations of the interpretation of this work, given that it was run on a subset of SNPs. However, our results demostrate the potential existence of much bigger groups of psoriasis-correlated alleles than was found by BlocBuster, despite the seemingly low odds-ratios. The possible existence of such clusters, combined with the qualitative differences between clusters found by BFS versus other methods considered here warrant a fuller investigation into such an approach at scale. This naturally comes with computational challenges, and clustering methods must be modified and adapted to graphs with billions of edges.

Of the methods considered, we find MCL and AP to be the most promising in terms of cluster results. Both these methods found fairly large clusters with a majority of SNPs with known psoriasis-associations. While commDet finds much smaller clusters on average, most of these clusters contain psoriasis-correlated alleles, and it is the fastest running method considered, which could become a factor with networks larger by orders of magnitude. Finally, we would consider extensive statistical significance testing in the future, when working with a more complete dataset and computational resources at hand.

7 CODE AVAILABILITY

All code generated in the course of this study is available in a publicly accessible repository at https://github.com/sayali7242/Masters-Project.

8 ACKNOWLEDGMENTS

I would like to thank Dr. Weixiong Zhang for his constant support and mentorship over the course of this project. I would also like to thank my committee members, Dr. Chenyang Lu and Dr. Netanel Raviv for their valuable inputs and thoughtful comments on my work. Finally, I would also like to thank Dr. Sharlee Climer for her help throughout the project.

Funding for the Collaborative Association Study of Psoriasis was provided by the National Institutes of Health, the Foundation for the National Institutes of Health, and the National Psoriasis Foundation. Support for genotyping of samples was provided through the Genetic Association Information Network (GAIN). The dataset(s) used for the analyses described in this manuscript were obtained from the database of Genotypes and Phenotypes (dbGaP) found at http://www.ncbi.nlm.nih.gov/gap through dbGaP accession number phs000019. Samples and associated phenotype data for the Collaborative Association Study of Psoriasis were provided by Drs. James T Elder (University of Michigan, Ann Arbor, MI), Gerald G Krueger (University of Utah, Salt Lake City, UT), Anne Bowcock (Washington University, St. Louis, MO) and GonÃğalo R Abecasis (University of Michigan, Ann Arbor, MI). For a description of the dataset, phenotypes, genotype data and quality control procedures see Nair et al (2009) Nature Genetics 41:200-204.

REFERENCES

[1] Haoyan Chen, Annie Poon, Celestine Yeung, Cynthia Helms, Jennifer Pons, Anne M Bowcock, Pui-Yan Kwok, and Wilson Liao. 2011. A genetic risk score combining ten psoriasis risk loci improves disease prediction. *PloS one* 6, 4 (2011), e19454.

[2] Sharlee Climer, Alan R Templeton, and Weixiong Zhang. 2014. Allele-specific network reveals combinatorial interaction that transcends small effects in psoriasis GWAS. *PLoS Comput Biol* 10, 9 (2014), e1003766.

[3] Sharlee Climer, Wei Yang, Lisa De Las Fuentes, Victor G Dávila-Román, and C Charles Gu. 2014. A custom correlation coefficient (CCC) approach for fast identification of multi-snp association patterns in genome-wide SNPs data. *Genetic epidemiology* 38, 7 (2014), 610–621.

[4] DL Duffy, LS Spelman, and NG Martin. 1993. Psoriasis in Australian twins. *Journal of the American Academy of Dermatology* 29, 3 (1993), 428–434.

[5] Anton J Enright, Stijn Van Dongen, and Christos A Ouzounis. 2002. An efficient algorithm for large-scale detection of protein families. *Nucleic acids research* 30, 7 (2002), 1575–1584.

[6] Brendan J Frey and Delbert Dueck. 2007. Clustering by passing messages between data points. *science* 315, 5814 (2007), 972–976.

[7] Teri A Manolio, Laura Lyman Rodriguez, Lisa Brooks, Gonçalo Abecasis, GAIN Collaborative Research Group, et al. 2007. New models of collaboration in genome-wide association studies: the Genetic Association Information Network. *Nature genetics* 39, 9 (2007), 1045.

[8] Rajan P Nair, Philip E Stuart, Ioana Nistor, Ravi Hiremagalore, Nicholas VC Chia, Stefan Jenisch, Michael Weichenthal, Gonçalo R Abecasis, Henry W Lim, Enno Christophers, et al. 2006. Sequence and haplotype analysis supports HLA-C as the psoriasis susceptibility 1 gene. *The American Journal of Human Genetics* 78, 5 (2006), 827–851.

[9] Andrew Ng, Michael Jordan, and Yair Weiss. 2001. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems* 14 (2001), 849–856.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[11] Leonard Richardson. 2007. Beautiful soup documentation. *April* (2007).

[12] Elisha DO Roberson and Anne M Bowcock. 2010. Psoriasis genetics: breaking the barrier. *Trends in Genetics* 26, 9 (2010), 415–423.

[13] A Tiilikainen, A Lassus, J Karvonen, P Vartiainen, and M Julin. 1980. Psoriasis and HLA-Cw6. *British Journal of Dermatology* 102, 2 (1980), 179–184.

[14] Stijn Marinus Van Dongen. 2000. *Graph clustering by flow simulation.* Ph.D. Dissertation.

[15] Rajagopal Venkatesaramani and Yevgeniy Vorobeychik. 2018. Community detection by information flow simulation. *arXiv preprint arXiv:1805.04920* (2018).

[16] James Vlasblom and Shoshana J Wodak. 2009. Markov clustering versus affinity propagation for the partitioning of protein interaction graphs. *BMC bioinformatics* 10, 1 (2009), 1–14.