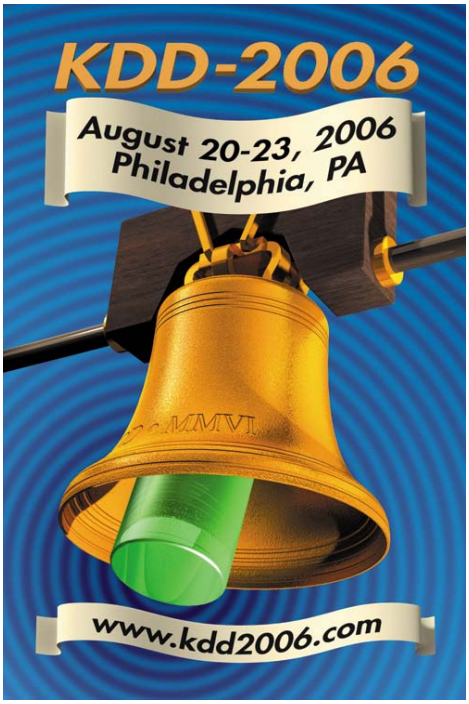
6th International Workshop on Data Mining in Bioinformatics (BIOKDD06)

Held in conjunction with the KDD conference, August 20, 2006



Workshop Chairs

Mohammed J. Zaki
George Karypis
Jiong Yang

BIOKDD06: Workshop on Data Mining in Bioinformatics
August 20th, 2006
Philadelphia, PA, USA

in conjunction with

12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining

Mohammed J. Zaki
Computer Science Department
Rensselaer Polytechnic Institute
Troy, NY 12180, USA
zaki@cs.rpi.edu

George Karypis
Department of Computer Science
University of Minnesota
Minneapolis, MN 55455, USA
karypis@cs.umn.edu

Jiong Yang
Electrical Engineering and Computer Science Department
Case Western Reserve University
Cleveland, OH 44106 USA
jiong@eecs.cwru.edu

Opening Remarks

Data Mining is the process of automatic discovery of novel and understandable models and patterns from large amounts of data. Bioinformatics is the science of storing, analyzing, and utilizing information from biological data such as sequences, molecules, gene expressions, and pathways. Development of novel data mining methods will play a fundamental role in understanding these rapidly expanding sources of biological data.

The goal of this workshop is to encourage KDD researchers to take on the numerous challenges that Bioinformatics offers. The workshop will feature invited talks from noted experts in the field, and the latest data mining research in bioinformatics. We encourage papers that propose novel data mining techniques for tasks like:

- Gene expression analysis
- Protein/RNA structure prediction
- Phylogenetics
- Sequence and structural motifs
- Genomics and Proteomics
- Gene finding
- Drug design
- RNAi and microRNA Analysis
- Text mining in bioinformatics
- Modeling of biochemical pathways

These proceedings contain 6 papers, out of 18 submissions, that were accepted for presentation at the workshop. Each paper was reviewed by three members of the program committee. Along with two keynote talks, by David Roos and Sridhar Hannenhalli, we were able to assemble a very exciting program.

We would like to thank all the authors, invited speaker, and attendees for contributing to the success of the workshop. Special thanks are due to the program committee for help in reviewing the submissions.

Workshop Co-Chairs

- Mohammed J. Zaki, Rensselaer Polytechnic Institute
- George Karypis, University of Minnesota
- Jiong Yang, Case Western Reserve University

Program Committee

• **Tatsuya Akutsu**, Kyoto University • **Chris Bailey-Kellogg**, Dartmouth College • **Mehmet Dalkilic**, Indiana University • **Yuan Gao**, Harvard University • **Shin Ishii**, Nara Institute of Science and Technology • **Sun Kim**, Indiana University • **Lei Liu**, University of Illinois, Urbana-Champaign • **Zoran Obradovic**, Temple University • **Gultekin Ozsoyoglu**, Case Western Reserve University • **Srinivasan Parthasarathy**, Ohio State University • **Jian Pei**, Simon Fraser University • **Naren Ramakrishnan**, Virginia Tech • **Hannu Toivonen**, University of Helsinki • **Jason Wang**, New Jersey Institute of Technology • **Wei Wang**, University of North Carolina, Chapel Hill • **Aidong Zhang**, SUNY, Buffalo • **Ying Zhao**, University of Missouri, Rolla

External Reviewers

• **Young-Rae Cho**, SUNY, Buffalo • **Woochang Hwang**, SUNY, Buffalo • **Mugdha Khaladkar**, New Jersey Institute of Technology • **Takeshi Kawabata**, Nara Institute of Science and Technology • **Mustafa Kirac**, Case Western Reserve University • **Chuan Lin**, SUNY, Buffalo • **Uros Midic**, Temple University • **Chandan Reddy**, Cornell University • **Juho Rousu**, University of Helsinki • **Yang Song**, New Jersey Institute of Technology • **Junilda Spirollari**, New Jersey Institute of Technology • **Dongrong Wen**, New Jersey Institute of Technology • **Hongbox Xie**, Temple University • **Wang Zhuo**, University of Illinois, Urbana-Champaign

Workshop Program & Table of Contents

8:50-9:00am: Opening Remarks. **page 1**

9:00-10:00am: Protein Interactions and Folding

- Signal Transduction Model Based Functional Module Detection Algorithm for Protein-Protein Interaction Networks, Woochang Hwang, Young-rae Cho, Aidong Zhang and Murali Ramanathan (State University of New York at Buffalo). **page 3**
- Protein Folding Trajectories: Summarization, Event Detection and Consensus Partial Folding Pathway Identification, Hui Yang, Srinivasan Parthasarathy and Duygu Ucar (Ohio State University). **page 13**

10:00-10:30am: Coffee Break

10:30-11:30am: Invited Talk

- David Roos, Merriam Professor of Biology & Director, Penn Genomics Institute, University of Pennsylvania. **page 22**

11:30-12:00pm: Cluster Visualization

- Automatic Layout and Visualization of Biclusters, Gregory A. Grothaus, Adeel Mufti and T. M. Murali (Virginia Polytechnic Institute and State University). **page 23**

12:00-1:30pm: Lunch

1:30-2:30pm: Invited Talk

- Deciphering Gene Regulatory Networks by in silico approaches, Sridhar Hannenhalli, Penn Center for Bioinformatics, Department of Genetics, University of Pennsylvania. **page 31**

2:30-3:30pm: Motif Discovery

- ExMotif: Efficient Structured Motif Extraction, Yongqiang Zhang and Mohammed Zaki (Rensselaer Polytechnic Institute). **page 32**
- Efficient Search using Hybrid EM for Motif Refinement, Chandan Reddy, Yao-Chung Weng and Hsiao-Dong Chiang (Cornell University). **page 42**

3:30-4:00pm: Coffee Break

2:30-3:30pm: Classification

- Protein Classification using Summaries of Profile-Based Frequency Matrices, Keith Marsolo and Srinivasan Parthasarathy (Ohio State University). **page 51**

Signal Transduction Model Based Functional Module Detection Algorithm For Protein-Protein Interaction Network

Woochang Hwang† Young-Rae Cho† Aidong Zhang† Murali Ramanathan††

†Department of Computer Science and Engineering, State University of New York at Buffalo,USA

††Department of Pharmaceutical Sciences, State University of New York at Buffalo,USA

Email: {whwang2, ycho8, azhang}@cse.buffalo.edu, murali@acsu.buffalo.edu

ABSTRACT

Cellular functions are coordinately carried out by groups of genes and proteins forming functional modules. Detection of such functional modules from protein-protein interaction (PPI) networks is one of the most challenging and important problem in post genomic era. Moreover, the sparse connectivity of protein-protein interaction data sets makes identification of functional modules more challenging. After careful observations of the properties of functional modules in the PPI network, we have found that the actual topological shapes and properties, including the graph density and diameter, of the functional modules in the PPI network have exposed unexpected phenomena, e.g., low intraconnectivity and longish shapes. Many different clustering approaches have been proposed to extract functional modules from PPI networks. Most of them concentrated only on densely connected regions topologically and ignored biological characteristics of the network to be dealt with, even though they were working on biological networks. Therefore, they could find only the clusters with certain density, and failed to find effective functional modules which are biologically significant. Furthermore, they produced many small size clusters, which have less than 5 members or even singletons, and it resulted in discarding huge number of proteins during the clustering process. To conquer these problems effectively, we develop an algorithm, termed STM, which utilizes the degree of influence between proteins to determine the cluster representatives. Clusters can then be formulated by an iterative merging process. STM is compared to six competing approaches including the maximum clique, quasi-clique, minimum cut, betweeness cut and Markov Clustering(MCL) algorithms. The clusters obtained by each technique are compared for enrichment of biological function. Identified clusters by STM are shown to be enriched for biological function better than the clusters identified by other existing approaches. Topological evaluation of the identified clusters by our method demonstrated that our method can successfully identify arbitrary shape clusters with large size that the other methods cannot. In addition to the above, an important strength of our approach is that the percentage of proteins that are discarded to create clusters is much lower than the other approaches which have an average discard percentage of 59% on the yeast protein-protein interaction network.

Keywords

Signal Transduction, Protein-protein interaction network, Functional module detection

1. INTRODUCTION

Since the first complete genome was sequenced in 1995, more than 300 prokaryotic genomes and more than 20 eukaryotic genomes have been sequenced[17]. Discovering the functional roles of gene products after the completion of sequencing the Saccharomyces Cerevisiae genome has been in the spotlight of post-genomic era. High-throughput techniques[5; 11; 12; 25] for protein-protein interactions (PPI) detection have attracted researchers' attention since interacting proteins are likely to serve together as a group in cellular functions[10]. In recent years, high-throughput techniques in a genomic scale such as yeast-two-hybrid, mass spectrometry, and protein chip technologies have multiplied the volume of protein interaction datasets exponentially and also have provided us a genomic level view of molecular interactions. The cumulative PPI dataset of, for example, S. Cerevisiae in DIP (Database of Interacting Proteins) [2] now lists over 4900 proteins and 18,000 interactions from over 22000 experiments; however, nearly half of the proteins remain unannotated. Effective computational systems for storage, management, visualization and analysis are necessary to cope with these fast growing complex datasets.

PPI data provide us the good opportunity to systematically analyze the structure of a large living system and also allow us to use it to understand essential principles like essentiality, genetic interactions, functions, functional modules, protein complexes and cellular pathways. Cellular functions and biochemical events are coordinately carried out by groups of proteins interacting each other in functional modules, and the modular structure of complex networks is critical to function[7; 10; 20]. Identifying such functional modules in PPI networks is very important for understanding the structure and function of these fundamental cellular networks. Therefore, developing an effective computational approach to identify functional modules should be highly challenging but indispensable. Clustering analysis helps us understand the topological structure of the PPI networks and relationships among its components better. And, the principal function of each cluster can be inferred from the functions of its member. Functions for unannotated members of a cluster can be predicted by the functions of other annotated members[18].

PPI adjacency matrices can be represented as graphs whose nodes represent proteins and edges represent interactions. The clustering of a PPI dataset can be thereby reduced to graph theoretical problems. But, the binary nature of the current PPI data sets imposes challenges in clustering using conventional approaches. In

the maximal clique approach, clustering is reduced to identifying fully connected subgraphs in the graph [24]. To overcome the relatively high stringency imposed by the maximal clique method, the Quasi Clique [2], Molecular Complex Detection (MCODE) [1], Spirin and Mirny [24] algorithms identify densely connected subgraphs rather than fully connected ones by either optimizing an objective density function or using a density threshold. The Restricted Neighborhood Search Clustering Algorithm (RNSC) [16] and Highly Connected Subgraphs (HCS) algorithms [9] harness minimum cost edge cuts for cluster identification. The Markov Cluster Algorithm (MCL) algorithm finds clusters using iterative rounds of expansion and inflation that promote the strongly connected regions and weaken the sparsely connected regions, respectively [26]. The line graph generation approach [20] transforms the network of proteins connected by interactions into a network of connected interactions and then uses the MCL algorithm to cluster the interaction network. Samantha and Liang [22] employed a statistical approach to clustering of proteins based on the premise that a pair of proteins sharing a significantly larger number of common neighbors will have high functional similarity.

However, currently used approaches encounter challenges because the relationship between protein function and PPI is characterized by weak connectivity and unexpected topological phenomena, such as low intraconnectivity and longish shapes of actual topological shapes of MIPS functional categories [19]. In our experimental analysis, subgraphs of each functional categories in MIPS database [19] are extracted from the Yeast PPI network, and the density of each subgraph is measured by Equation 7. The density of those subgraphs is averaged about 0.0023 which is fairly lower than we expected. Most functional categories have low connectivity within them in the PPI network and the majority of members in functional categories do not have direct physical interaction with other members of the functional category they belong to. Furthermore, it is not difficult to find singletons in the extracted subgraphs of functional categories which means that some proteins do not have any interaction with other proteins in the functional category they belong to. Let the average diameter of a graph be the length of the longest path among all pair shortest paths in the graph. The average diameter of the subgraphs of all MIPS functional categories is approximately 4 which is close to the average shortest paths length, 5.47, of the whole PPI network. In other words, the subgraphs of actual MIPS functional categories in the PPI network generally are not closely congregated as we expected, they have longish shapes. Due to such low density within the modules, the existing approaches have produced a number of clusters with small size and singletons and mercilessly discarded many weakly connected nodes since they can only handle the strongly connected regions. Such incompleteness of clustering is a serious drawback of the existing algorithms. Discarding the sparsely connected nodes could be a hazardous loss of important information of the PPI network.

Biological networks, including PPI networks, illustrate the biochemical relationships of components in biochemical processes. Clustering of biological networks should be able to identify clusters of any arbitrary shapes and any density if the members of a cluster share important biochemical properties from the point of view of biochemical processes. To cope with this necessity and overcome those drawbacks of existing approaches, we propose a novel strategy to analyze the degree of biological and topological influence of each protein to other proteins in a PPI network. We model PPI networks as a dynamic signal transduction system and demonstrate the signal transduction behavior of the perturbation by each protein on PPI networks. This behavior should also reflect the topological properties of the network. The overall signal transduction behavior function between any two proteins will be formulated to evaluate the perturbation caused by a protein on other proteins biologically and topologically in the network. STM successfully identified the clusters with bigger size, arbitrary shape, low density, and biologically more enriched than other existing approaches did.

2. METHOD

2.1 The Signal Transduction Model

The proteins and the protein-protein interactions in a PPI data set were, respectively, represented by nodes and edges of a graph. The graph representation was modeled using a pharmacodynamic signal transduction network model. Specifically, the signal transduction behavior of the network was modeled using the Erlang distribution, a special case of the Gamma distribution.

Graph definitions: An undirected graph $G = (V, E)$ consists of a set V of nodes and a set E of edges, $E \subseteq V \times V$. An edge $e = (i, j)$ connects two nodes i and j, $e \in E$. The neighbors $N(i)$ of node i are defined to be the set of directly connected nodes of node i. The degree $d(i)$ of a node i is the number of the edges connected to node i. A path is defined as a sequence of nodes $(i_1, \ldots, i_k)$ such that from each of its nodes there is an edge to the successor node. The length of a path is the number of nodes in its node sequence. A shortest path between two nodes, i and j, is a minimal length path between them. The distance between two nodes, i and j, is the length of its shortest path.

Signal transduction model: We propose to model the dynamic relationships between proteins in a PPI network using a signal transduction network model. Specifically, the signal transduction behavior of the network is modeled using the Erlang distribution, a special case of the Gamma distribution. The Erlang distribution function is:

$$F(x) = 1 - e^{-\frac{x}{b}} \times \sum_{k=0}^{c-1} \frac{(\frac{x}{b})^k}{k!} \quad (1)$$

where $c > 0$ is the shape parameter, $b > 0$ is the scale parameter, $x \geq 0$ is the independent variable, usually time. The Erlang distribution has several characteristics, which are appropriate for describing the protein-protein interaction network, including its positive range and its important reproductive property [14]. The Erlang distribution with $x/b = 1$ is used and the value of c is set to the number of edges between source protein node and the target protein node. Setting the value of x/b to unity assesses the perturbation at the target protein when the perturbation reaches $1/e$ of its initial value at the nearest neighbor of the source protein node.

Erlang distribution models have been used in pharmacodynamics to model signal transduction and transfer delays in a variety of systems including the production of drug induced mRNA and protein dynamics [21] and calcium ion-mediated signaling in neutrophils [8]. Figure 1 shows the corresponding pharmacodynamic signal transduction model whose bolus response is an Erlang distribution with parameters b and c, respectively. Statistically, the Erlang distribution represents the time required to carry out a sequence of c tasks whose durations are identical, exponential probability distributions. The Gamma distribution is also excellent for approximating population abundances fluctuating around equilibrium [4]. Thus in the context of the PPI network, the Erlang distribution can be viewed as a mechanistically motivated measure of the perturbation caused by the sudden introduction of the source protein to a target protein in the network.

The Erlang distribution needs to be further modified to reflect network topology. The perturbation induced by the source protein

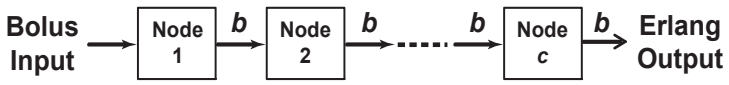


Figure 1: The pharmacodynamic signal transduction model whose bolus response is an Erlang distribution. The b is the time constant for signal transfer and c is the number of compartments.

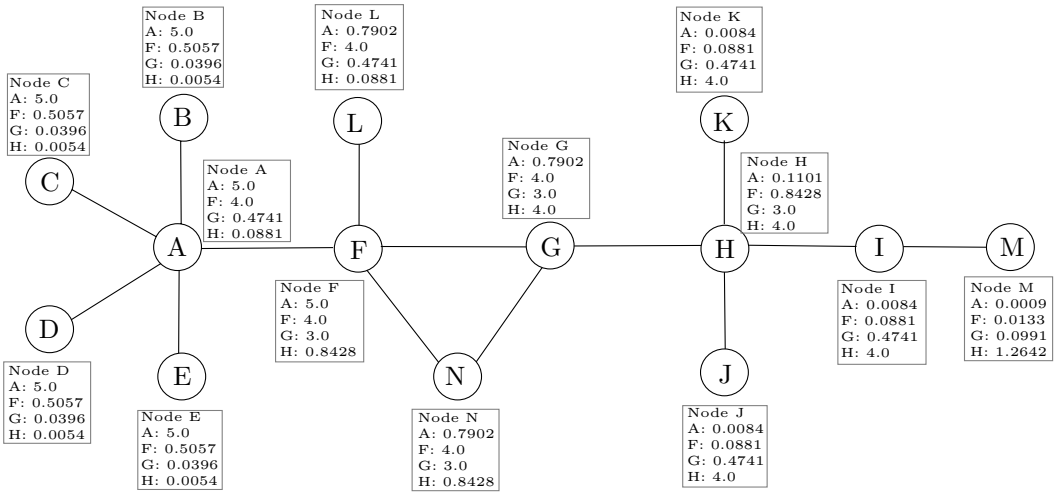


Figure 2: A simple network. Each box contains the numerical values obtained from Equation 2 from nodes A, F, G, and H to other target nodes. Results for other nodes are not shown.

node should be proportional to its degree and to follow the shortest path to the target protein node. During transduction to the target protein node, the perturbation should dissipate at each intermediate visiting node to each incident edge. The overall signal transduction behavior function from node v to node w ($v \neq w$) is thus:

$$S(v \rightarrow w) = \frac{d(v)}{\prod_{i \in P(v,w)} d(i)} \times F(x) \qquad (2)$$

where $d(i)$ is the degree of node i, $P(v, w)$ is the set of the all nodes visited en route from node v to node w, excluding the source node v and the target destination node w, and $F(x)$ is the signal transduction behavior function. When $v = w$ and $distance(v, w) = 0$, we define $S(v \rightarrow w) = d(v)$. The numerator of the first term in the right hand side of Equation 2 represents the degree of the source node v, and the denominator represents the dissipation on each visiting node on the shortest path from source node v to target node w. Our choice of the shortest path is motivated by the finding that the majority of flux prefers the path of least resistance in many physicochemical and biological systems. So, the first term in the right hand side of Equation 2 represents the topological effect of source node v on target node w. The second term in the right hand side of Equation 2 represents the biological effect of source node v on target node w in the signal transduction view point. Therefore, the nodes that score the highest value on target node w will be the most influential nodes on node w biologically and topologically.

Figure 2 demonstrates the signal transduction behavior of a small example network according to Equation 2. For the ease of understanding, only the signals from node A, F, G, and H are presented, although signals should be propagated from each node in the network. Each box in Figure 2 contains the signal assessed by the Equation 2 from nodes A, F, G, and H to other target nodes, e.g., 5.0, 0.5057, 0.0396, 0.0054 are the signals assessed from nodes A, F, G, and H, respectively, on node E. These numerical values illustrate overall effects of combining the network topology with the signal transduction model from source nodes A, F, G, and H on node E. Consequently, node A, which has scored the highest value, will be the most influential node on node E biologically and topologically.

2.2 Clustering Model

STM algorithm simulates the perturbation from each node to the other nodes in a network using Equation 2, which reflects the biological and topological properties of the node. Module representatives are the nodes that record the highest scores by Equation 2 on every node in a module, i.e., they are the most influential nodes in a module biologically and topologically. After the signal transduction simulation, each node selects the most influential nodes as the representatives of modules. From these representatives, preliminary modules can be formed by aggregating each node into each module that each of its representatives stands for. Finally, these preliminary modules are merged if there are substantial interconnections between them.

The pseudocode for the STM algorithm, which employs the signal transduction function of Equation 2 and a democratic representatives selection algorithm, is shown in Algorithm 1. The algorithm involves four sequential processes:

Process 1: Compute signals transduced between all node pairs.
Process 2: Select cluster representatives for each node.
Process 3: Formation of preliminary clusters.
Process 4: Merge preliminary clusters.

Process 1 propagates signals from each source node and records the signal quantities on each target node for all node pairs according to Equation 2. The implementation of Process 1 is shown on lines 6-10 of the STM algorithm in Algorithm 1.

In Process 2, each node elects the nodes from which it receives the highest signal value as the representatives of the clusters that the node will belong to. For example, in Figure 2, nodes A, B,

Algorithm 1 STM(G)

1: V: set of nodes in graph G
2: $F(x)$: Transduction behavior function
3: $S(v, w)$: arrived signal from node v to node w
4: C: the list of final clusters
5: PreClusters: the list of preliminary clusters
6: **for** *each node pair*(v, w) $v, w \in V, v \neq w$ **do**
7: $distance(v, w) \leftarrow$ the shortest path length from node v to node w
8: set parameter c in function as $distance(v, w)$
9: $signal(v, w) \leftarrow S(v \rightarrow w)$
10: **end for**
11: **for** *each node* $v \in V$ **do**
12: $v.representative \leftarrow$ select the best scored node w for node v
13: **if** $cluster\_w == null$ **then**
14: make $cluster\_w$
15: $cluster\_w.add(v)$
16: $PreClusters.add(cluster\_w)$
17: **else**
18: $cluster\_w.add(v)$
19: **end if**
20: **end for**
21: C $\leftarrow$ **Merge**(PreClusters)

Procedure 1 Merge(C)

1: C: the cluster list
2: MaxPair: the cluster pair(c, k) with max interconnections among all pairs
3: Max.value: interconnections between cluster pair c and k
4: MaxPair $\leftarrow$ findMaxPair(C,null)
5: **while** Max.value$\geq$threshold **do**
6: newCluster $\leftarrow$ merge MaxPair c and k
7: Replace cluster c with newCluster
8: Remove cluster k
9: MaxPair $\leftarrow$ findMaxPair(C,newCluster)
10: **end while**
11: **return** C

C, D, E, and F will choose node A and nodes L, G, and N will choose node F, which are the best scored nodes on those nodes, as the representatives.

Each preliminary cluster is initialized by taking its representative as its initial member. Preliminary clusters are then augmented by accumulating each node toward each cluster that each of the representatives, which are chosen by the node, stands for. Lines from 11-20 in Algorithm 1 contain the representative selection process and the preliminary cluster formation process. Notice that STM allows overlaps among clusters by opening the possibility of multiple representatives which have the tie score on a node, etc. For example, node G picks nodes F and H, which have the tie score on node G, as its representatives in Figure 2. Then, G will belong to the cluster formed by nodes F and the cluster formed by H. Therefore, overlaps occur between the cluster formed by node F, {F, G, L, N}, and the cluster formed by node H, {G, H, I, J, K, M}. STM identified three preliminary clusters, {A, B, C, D, E, F}, {F, G, L, N}, and {G, H, I, J, K, M}, based on the choice of representatives in Figure 2.

Some preliminary clusters may be merged if they have substantial interconnections. We propose to measure the degree of interconnectivity between clusters by the similarity of two clusters i and j defined below:

$$Similarity(i, j) = \frac{interconnectivity(i, j)}{minsize(i, j)} \tag{3}$$

where $interconnectivity(i, j)$ is the number of connections between clusters i and j, and $minsize(i, j)$ is the size of the smaller cluster among clusters i and j. The $Similarity(i, j)$ between two clusters i and j is the ratio of the number of the connections between them to the size of the smaller cluster. Highly interconnected clusters are iteratively merged based on the similarity of the clusters. The pair of clusters that have the highest similarity are merged in each iteration and the merge process iterates until the highest similarity of all cluster pairs is less than a given threshold. The selection of the threshold for merging clusters is a critical factor for the final cluster outcome. Theoretically, we can see when $interconnectivity(i, j) \geq minsize(i, j)$, clusters i and j have substantial interconnections. Three clusters, {A, B, C, D, E, F}, {F, G, L, N}, {G, H, I, J, K, M}, are obtained after the Process 4 when 2.0 is used as the merge threshold. Two clusters, {A, B, C, D, E, F, G, L, N}, {G, H, I, J, K, M}, are obtained after the Merge process when 1.0 is used as the merge threshold.

2.3 Cluster Assessment

The structures of the clusters identified by STM and other competing alternative approaches are assessed using several metrics.

The clustering coefficient, $C(v)$, of a node v measures the connectivity among its direct neighbors:

$$C(v) = \frac{2 \mid \bigcup_{i,j \in N(v)} (i, j) \mid}{d(v)(d(v) - 1)} \tag{4}$$

In Equation 4, $N(v)$ is the set of the direct neighbors of node v and $d(v)$ is the number of the direct neighbors of node v. Highly connected nodes have high values of clustering coefficient.

Degree centrality orders nodes by the number of their direct neighbors, and betweenness centrality measures the nodes' importance from the information flow point of view in a network. Degree and betweenness centrality commonly used to measure the importance of a node in a network. The Betweeness Centrality, $C_B(v)$, is a measure of the global importance of a node that assesses the proportion of shortest paths between all node pairs that pass through the node of interest. The Betweeness Centrality, $C_B(v)$ for a node of interest, v, is defined by:

$$C_B(v) = \sum_{s \neq v \neq t \in V} \frac{\rho_{st}(v)}{\rho_{st}} \tag{5}$$

In the Equation 5, ρ_{st} is the number of shortest paths from node s to t and $\rho_{st}(v)$ the number of shortest paths from s to t that pass through the node v.

The extent to which the clusters are associated with a specific biological function is evaluated using a p-value based on the hypergeometric distribution [2]. The p-value is the probability that a cluster would be enriched with proteins with a particular function by chance alone. The p-value is given by:

$$p = 1 - \sum_{i=0}^{k-1} \frac{\binom{C}{i}\binom{G-C}{n-i}}{\binom{G}{n}} \tag{6}$$

In Equation 6, C is the size of the cluster containing k proteins with a given function; G is the size of the universal set of proteins of known proteins and contains n proteins with the function. Because the p-values are frequently small numbers with positive val-

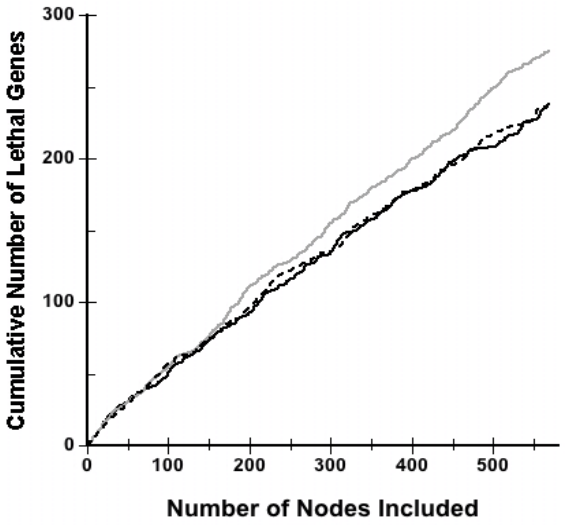


Figure 3: Accumulation of lethal proteins for various percentiles of degree (gray line), betweeness centrality (dashed line) or the STM signal transduction metric (solid line). The results are shown for the top 555 proteins obtained from the yeast PPI network and are ordered; the highest values of these metrics are closest to the origin.

ues between 0 and 1, the negative logarithms (to base 10, denoted -log p) are used. A -log p value of 2 or greater indicates statistical significance at $\alpha = 0.01$.

The density of a subgraph s in a PPI network is measured by:

$$D_s = \frac{2e}{n(n-1)} \tag{7}$$

In Equation 7, n is the number of proteins and e is the number of interactions in a subgraph s of a PPI network.

3. EXPERIMENTAL RESULTS

3.1 Protein Interaction Data

The core data of S. Cerevisiae was obtained from the DIP database [3]. This dataset include 2526 proteins and 5949 filtered reliable physical interactions. Species such as S. Cerevisae provide important test beds for the study of the PPI networks since it is a well-studied organism for which most proteomics data is available for the organism, by virtue of the availability of a defined and relatively stable proteome, full genome clone libraries, established molecular biology experimental techniques and an assortment of well designed genomics databases [3; 10].

3.2 Biological Significance of the Putative Module Representatives

Our signal transduction model of Equation 2 provides a vehicle to quantitatively measure the degree of biological and topological influence of each protein on other proteins in the PPI network. The most influential proteins, that is, the highest scored nodes, are highly important proteins. To evaluate the biological significance of the most influential proteins, we annotated the lethality of each protein in the yeast PPI network according to the MIPS lethality data. Lethality is a crucial factor to characterize the biological essentiality of a protein. It is determined by examining whether a module is functionally disrupted when the protein is knocked out. We obtained the protein lethality information from MIPS database [19], which reports whether a protein is lethal or viable. We found that 233 proteins out of the top scored 555 proteins are lethal.

Figure 3 plots the cumulative number of lethal genes vs. the number of protein nodes included for increasing percentiles of the degree, betweeness or the STM signal transduction metric. The data are shown for 555 genes, obtained from the yeast PPI network, with the highest values of each of these metrics. In each case, the results are sorted and highest values are placed closest to the origin. Figure 3 shows that the performance of the STM metric in predicting lethality is comparable to that of degree and betweeness approaches for up to 150 nodes.

3.3 Clustering Performance Analysis

Experimentally, we performed STM algorithm on the yeast PPI data set using various merge threshold values to find the best threshold value for each data set. Experiments using 0.5, 1.0, 1.5, 2.0, 2.5, and 3.0 as the merge threshold were performed on each data set. The results show that when the merge threshold is less than 1.0, clusters that do not have substantial similarity are merged; and when the merge threshold is greater that 1.5, merging seldom occurred. There is no much performance difference when the values between 1.0 and 1.5 are used. The experiment when 1.0 is used as the merge threshold showed the best performance.

3.3.1 Cluster Analysis

555 preliminary clusters are obtained from the yeast PPI network and merged using 1.0 as the merge threshold. In Table 1, all 60 clusters that have more than 4 proteins are listed, and it also shows their topological characteristics and their assigned molecular functions from MIPS functional categories. To facilitate critical assessments, the percentage of proteins that are in concordance with the major assigned function (hits), the discordant proteins (misses) and un-known are also indicated. Among these 60 clusters, the largest one contains 210 proteins and the smallest one contains 5 in them. On average, we have 40.1 proteins in a cluster, and the average density of the subgraphs of the clusters extracted from the PPI network is 0.2145. The -log p values of the major function identified in each cluster is also shown and these values provide a measure of the relative enrichment of a cluster for a given functional category: higher values of -log p indicate greater enrichment. The results demonstrate that the STM method can detect large but sparsely connected clusters as well as small densely connected clusters. The high values of -log p (values greater than 2.0 indicate statistical significance at $\alpha < 0.01$) indicate that clusters are significantly enriched for biological function and can be considered to be functional modules. As a result, our method can clearly identify larger modules that have low density but still biologically enriched as we can see from the size, the density, and the P-value of the clusters in Table 1.

Figure 4 exhibits the distribution of the hit, miss, and unknown percentage of member proteins with the assigned function for each cluster in Table 1 for better understanding visually. We found that most of the proteins in a cluster have the same functions that are assigned as a main function for the cluster as shown in Figure 4.

3.3.2 Comparative Analysis

The results in Table 2 and 3 for the yeast PPI dataset show that STM generates larger clusters; the clusters identified had p-values that are 2.2 orders of magnitude or approximately 125-fold lower than Quasi clique, the best performing alternative clustering method, on biological function. The p-values for the cellular localization are also shown in the last column of Table 2 and 3. It is clear that the clusters identified by STM despite being larger have low p-values. Although p-values generally decrease with increasing cluster size,

Cluster	Size	Density	Distribution			-Log*p*	Function
			H	D	U		
1	214	0.019	24.7	69.6	5.6	43.9	Nuclear transport
2	188	0.015	69.1	25.0	5.8	36.4	Cell cycle and DNA processing
3	181	0.022	22.0	72.3	5.5	17.2	Cytoplasmic and nuclear protein degradation
4	170	0.028	46.4	42.9	10.5	31.6	Transported compounds (substrates)
5	131	0.028	37.4	55.7	6.8	28.6	Vesicular transport (Golgi network, etc.)
6	125	0.030	60.8	33.6	5.6	32.2	tRNA synthesis
7	113	0.027	19.4	71.6	8.8	11.8	Actin cytoskeleton
8	79	0.045	17.7	73.4	8.8	12.3	Homeostasis of protons
9	78	0.033	26.9	62.8	10.2	12.5	Ribosome biogenesis
10	76	0.041	38.1	59.2	2.6	20.2	rRNA processing
11	72	0.030	5.6	84.7	9.7	6.2	Calcium binding
12	68	0.064	66.1	25.0	8.8	44.5	mRNA processing
13	61	0.041	40.9	52.4	6.5	11.5	Cytoskeleton
14	58	0.064	72.4	27.6	0.0	37.4	General transcription activities
15	53	0.048	15.0	71.6	13.2	7.9	MAPKKK cascade
16	50	0.064	66.0	32.0	2.0	33.5	rRNA processing
17	45	0.055	24.4	73.3	2.2	11.1	Metabolism of energy reserves
18	44	0.058	59.0	36.3	4.5	5.1	Metabolism
19	39	0.072	10.2	89.7	0.0	7.3	Cell-cell adhesion
20	36	0.125	58.3	36.1	5.5	16.9	Vesicular transport
21	29	0.091	55.1	44.8	0.0	8.3	Phosphate metabolism
22	28	0.074	14.2	78.5	7.1	4.5	Lysosomal and vacuolar protein degradation
23	27	0.119	29.6	66.6	3.7	7.3	Cytokinesis (cell division)/septum formation
24	26	0.153	53.8	46.1	0.0	28.6	Peroxisomal transport
25	25	0.090	28.0	68.0	4.0	4.6	Regulation of C-compound and carbohydrate utilization
26	25	0.116	68.0	28	4.0	12.9	Cell fate
27	22	0.151	59.0	36.3	4.5	11.4	DNA conformation modification
28	21	0.147	76.1	19.0	4.7	23.9	Mitochondrial transport
29	20	0.200	75.0	20.0	5.0	24.0	rRNA synthesis
30	19	0.228	78.9	15.7	5.2	17.9	Splicing
31	17	0.220	70.5	29.4	0.0	19.7	Microtubule cytoskeleton
32	17	0.183	23.5	76.4	0.0	8.2	Regulation of nitrogen utilization
33	15	0.304	86.6	13.3	0.0	31.3	Energy generation
34	14	0.142	50.0	42.8	7.1	9.0	Small GTPase mediated signal transduction
35	13	0.564	76.9	23.0	0.0	15.9	Mitosis
36	13	0.358	84.6	15.4	0.0	12.4	DNA conformation modification
37	13	0.410	69.2	23.0	7.6	17.6	3'-end processing
38	13	0.179	61.5	30.7	7.6	6.7	DNA recombination and DNA repair
39	12	0.196	16.6	75.0	8.3	3.9	Unspecified signal transduction
40	12	0.363	58.3	41.6	0.0	14.7	Posttranslational modification of amino acids
41	12	0.166	16.6	75.0	8.3	2.4	Autoproteolytic processing
42	11	0.218	54.5	45.4	0.0	2.9	Transcriptional control
43	11	0.200	72.7	27.2	0.0	8.2	Enzymatic activity regulation / enzyme regulator
44	10	0.466	80.0	20.0	0.0	14.8	Translation initiation
45	9	0.361	77.7	22.2	0.0	12.8	Translation initiation
46	8	0.321	50.0	37.5	12.5	5.6	Metabolism of energy reserves
47	8	0.321	75.0	25.0	0.0	9.0	Modification by ubiquitination, deubiquitination
48	8	0.321	37.5	62.5	0.0	3.7	Mitosis
49	7	0.333	42.8	57.1	0.0	3.5	DNA damage response
50	7	0.333	57.1	28.5	14.2	4.1	Vacuolar transport
51	7	0.285	28.5	71.4	0.0	4.4	Biosynthesis of serine
52	6	0.333	50.0	33.3	16.6	2.38	Modification by phosphorylation, dephosphorylation, etc.
53	5	0.400	100	0.0	0.0	7.0	Meiosis
54	5	0.600	100	0.0	0.0	7.0	Vacuolar transport
55	5	0.400	100	0.0	0.0	8.5	ER to Golgi transport
56	5	0.400	20.0	40.0	40.0	1.8	cAMP mediated signal transduction
57	5	0.500	40.0	40.0	20.0	3.1	Oxidative stress response
58	5	0.500	80.0	20.0	0.0	4.4	Intracellular signalling
59	5	0.600	40.0	60.0	0.0	4.2	Tetracyclic and pentacyclic triterpenes
60	5	0.400	60.0	40.0	0.0	4.1	Mitochondrial transport

Table 1: Clusters obtained using STM for the yeast PPI network. The first column is a cluster identifier; the Size column indicates the number of proteins in each cluster; the Density indicates the percentage of possible protein interactions that are present; the H column indicates the percentage of proteins concordant with the major function indicated in the last column; the D column indicates the percentage of proteins discordant with the major function and U column indicates percentage of proteins not assigned to any function. The -log p values for biological function are shown.

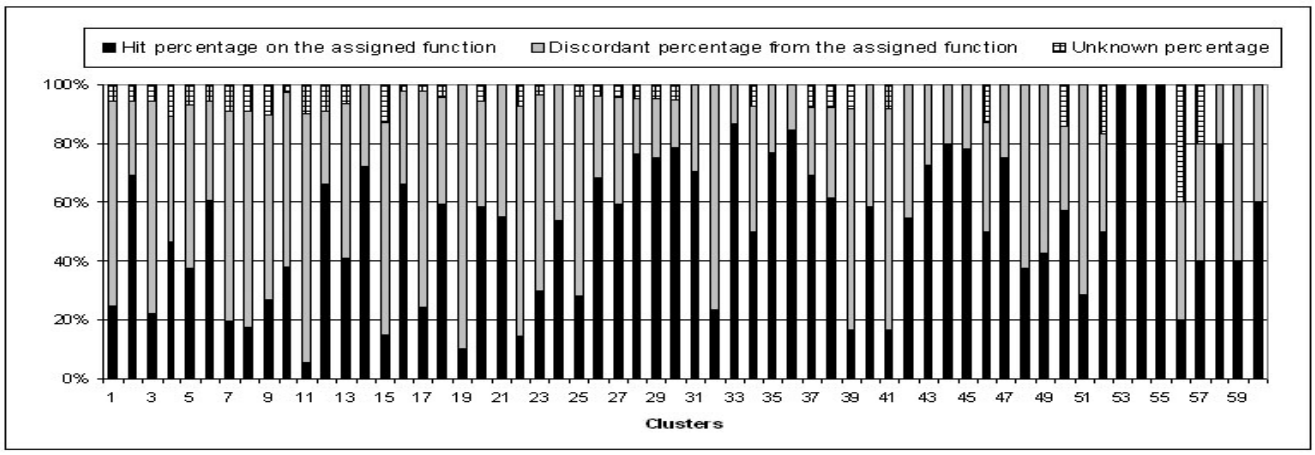


Figure 4: Distribution of the three classes of 60 clusters: the hit percentage with the assigned function, discordant percentage from the assigned function, and unknown percentage

these decreases in p-values can occur only when the null hypothesis is false. The p-values reflect the confidence that the differences, if present, are not due to chance alone. The confidence in any given result increases when these are obtained in a larger sample and in this context. So, the dependence of p-values on sample size is intuitive. The p-values express the strength of evidence against the null hypothesis to account for both the sample size, the amount of noise in measurements. Therefore, the STM clusters have low p-values because they are enriched for function and not simply because they are larger.

Tables 2 and 3 demonstrate that STM outperforms the other existing approaches. We made a comparison with 6 other existing approaches, Maximal cliques [24], Quasi cliques [2], Samantha [23], Minimum cut [14], Betweenness cut [6], and MCL [26]. The comparison on the cluster size more than 4 is in Table 2 and on the cluster size more than 9 in Table 3. Both tables show that our signal transduction model based method generates considerably larger clusters, and the identified clusters by our method have at least 2 orders of magnitude higher P-value than the others on both function and localization categories.

Quasi clique and Maximal clique discarded 80.8% and 98.4% nodes during clustering process, even though they identified the clusters with relatively high p-values in Table 2. Quasi clique and Samantha discarded 86.7% and 93.3% nodes, even though they identified the clusters with relatively high p-values in the clusters with size more than 9 in Table 3. Another important strength of STM is that the percentage of proteins that are discarded to create clusters is 7.8%, which is much lower than the other approaches, which have an average discard percentage of 59%. The yeast PPI dataset is relatively modular and the bottom-up approaches (e.g., maximal clique and quasi clique methods) generally outperformed the top-down approaches (exemplified by the minimum cut and betweeness cut methods) on functional enrichment as assessed by -log p. However because bottom-up approaches are based on connectivity of dense regions, the percentages of discarded nodes for the bottom-up methods are also higher than STM and the top-down approaches. But, we already have shown that the functional modules have fairly low density and arbitrary shapes with long diameter. So, discarding those sparsely connected proteins could be a fatal decision which might resulted in the important biological information losses. Consequently, STM is versatile and its performance on biological function and localization enrichment, cluster size, and discard rate is superior to the best of the other six methods on both data sets.

3.3.3 Topological Analysis

To analyze the clustering results visually, we observed the topological shapes of the detected clusters by STM and their associated functional categories that were assigned from MIPS database. 4 example clusters which have visually recognizable size are selected from the 60 clusters in Table 1. The subgraphs of these chosen clusters were extracted from the yeast PPI network and displayed on the left column on each row in Figure 5. The MIPS function categories that were assigned to each of these clusters were also extracted from the yeast PPI network and displayed on the right side of its associated cluster. Most of the detected clusters have similar backbone structures with their associated functions. Notice that we can easily find out that the diameter and the density of those example clusters and their assigned functions are fairly long and sparse. Also notice that it is not difficult to find singletons in each functional category subgraphs, which means that there are members which do not have a direct physical interaction within the functional category that they belong to.

3.4 Computational Complexity Analysis

STM is fundamentally established on all pairs shortest path searching algorithm to measure the distance between all node pairs. This problem can be solved in $O(V^2 logV + VE)$ time if it is implemented using Johnson's algorithm [13], where V is the number of nodes and E is the number of edges in a graph. After measuring the distance between all node pairs, formation of preliminary clusters takes $O(V)$ time. The amount of time required to find the best cluster pair that has the most interconnections is $O(k^2 logk)$ by using heap-based priority queue, where k is the number of preliminary clusters [15]. The Merging process needs to find the cluster pair which has the most interconnections, and it takes $O(k^2 logk)$ time only for the initial iteration. From the second iteration, finding the best cluster pair takes $O(klogk)$ time since the cluster pair comparisons are needed only between the newly merged cluster and the other clusters. And the maximum k, the number of preliminary clusters, is at most $O(V)$ in the case of the fully connected graph, therefore the Merging process takes $O(V^2 logV)$ time. But k is much smaller than V in sparse networks like the Yeast PPI network. So the total time complexity of our algorithm is bounded

Method	Number	Size	Discard(%)	Function	Location
STM	**60**	**40.1**	**7.8**	**13.7**	**7.42**
Maximal clique	120	5.65	98.4	10.6	7.93
Quasi clique	103	11.2	80.8	11.5	6.58
Samantha	64	7.9	79.9	9.16	4.89
Minimum cut	114	13.5	35.0	8.36	4.75
Bwtweenness cut	180	10.26	21.0	8.19	4.18
MCL	163	9.79	36.7	8.18	3.97

Table 2: Comparison of STM to competing clustering methods for the yeast protein-protein interaction data set for clusters with 5 or more members. The Number column indicates the number of clusters identified by each method, the Size column indicates the average number of proteins in each cluster; the Discard% indicates the percentage of proteins not assigned to any cluster. The -log p values for biological function and cellular location are shown.

Method	Number	Size	Discard(%)	Function	Location
STM	**45**	**52.4**	**11.5**	**16.8**	**9.01**
Maximal clique	N/A	N/A	N/A	N/A	N/A
Quasi clique	46	16.7	86.7	15.3	9.34
Samantha	17	12.3	93.3	15.9	7.65
Minimum cut	44	24.3	55.0	14.8	8.78
Bwtweenness cut	78	14.4	50.5	11.3	6.05
MCL	55	16.7	69.4	11.5	5.42

Table 3: Comparison of STM to competing clustering methods for the yeast protein-protein interaction data set for clusters with 9 or more members. The Maximal clique does not identify clusters with 9 or more members. The footnote is the same to Table 2.

by the time consumed in computing the distance between all node pairs, which is $O(V^2 logV + VE)$.

4. DISCUSSION

We have studied that the topological shapes of the subgraphs of MIPS functional categories extracted from the PPI network are arbitrary and the density of them is fairly low. These two unexpected properties of functional categories prohibited other existing approaches from detecting functional modules from PPI networks effectively. A relative excess of emphasis on density and interconnectivity in the existing methods can be preferential for detecting clusters with relatively balanced round shapes and limit performance. The incompleteness of clustering is another distinct drawback of existing algorithms, which produce many clusters with small size and singletons. The preference for strongly connected nodes results in many weakly connected nodes being discarded. Moreover, considering only the topological properties and ignoring the biological characteristics of the network also can damage the effectiveness of clustering.

In this paper we have proposed a novel clustering method based on the signal transduction model for the Yeast PPI network. In head-to-head comparisons, the STM outperformed competing approaches and is capable of effectively detecting both dense and sparsely connected, biologically relevant functional modules with fewer discards. To our knowledge, this is the first description of the use of signal transduction based approach for this application.

Overwhelming performance of our approach has been demonstrated in several criteria including visual inspection. STM generated bigger size clusters with arbitrary shape, and those identified clusters are more biologically enriched, i.e., higher P-value, even though they have low density. There are more than 5% of unannotated proteins in the identified clusters. The function of those unannotated proteins can be predicted according to their assigned main functions by our method. Completeness of our clustering method is another distinct strength compared to the other methods. Our method discarded only about 7.8% of proteins which is tremendously lower than the other approaches did, 59% in average. In conclusion, STM has strong pharmacodynamics-based underpinnings and is an effective, versatile approach for analyzing protein-protein interactions. The STM approach contains a framework for rationally incorporating reaction rates, protein concentrations and interaction stoichiometry should these become available. It could therefore have potential applications in the drug discovery and development.

5. REFERENCES

[1] Bader, G.D. and Hogue. C.W. An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinformatics*, 4:2, 2003.

[2] Bu, D. et al. Topological structure analysis of the protein-protein interaction network in budding yeast. *Nucleic Acids Res.*, 31:2443–2450, 2003.

[3] Deane, C.M., Salwinski, L., Xenarios, I. and Eisenberg, D. . Protein interactions: two methods for assessment of the reliability of high throughput observations. *Mol. Cell Proteomics*, 1:349–356, 2002.

[4] Dennis, B. and Patil, G.P. . The gamma distribution and weighted multimodal gamma distributions as models of population abundance. *Math. Biosciences*, 68:187–212, 1984.

[5] Gavin, A.C. et al. . Functional organization of the yeast proteome by systematic analysis of protein complexes. *Nature*, 415:141–147, 2002.

[6] Girvan, M. and Newman, M. E. J. Community structure in social and biological networks. *PNAS*, 99:7821–7826, 2002.

[7] Guimera R. and Nunes Amaral L. Functional cartography of complex metabolic networks. *Nature*, 433:895–900, 2005.

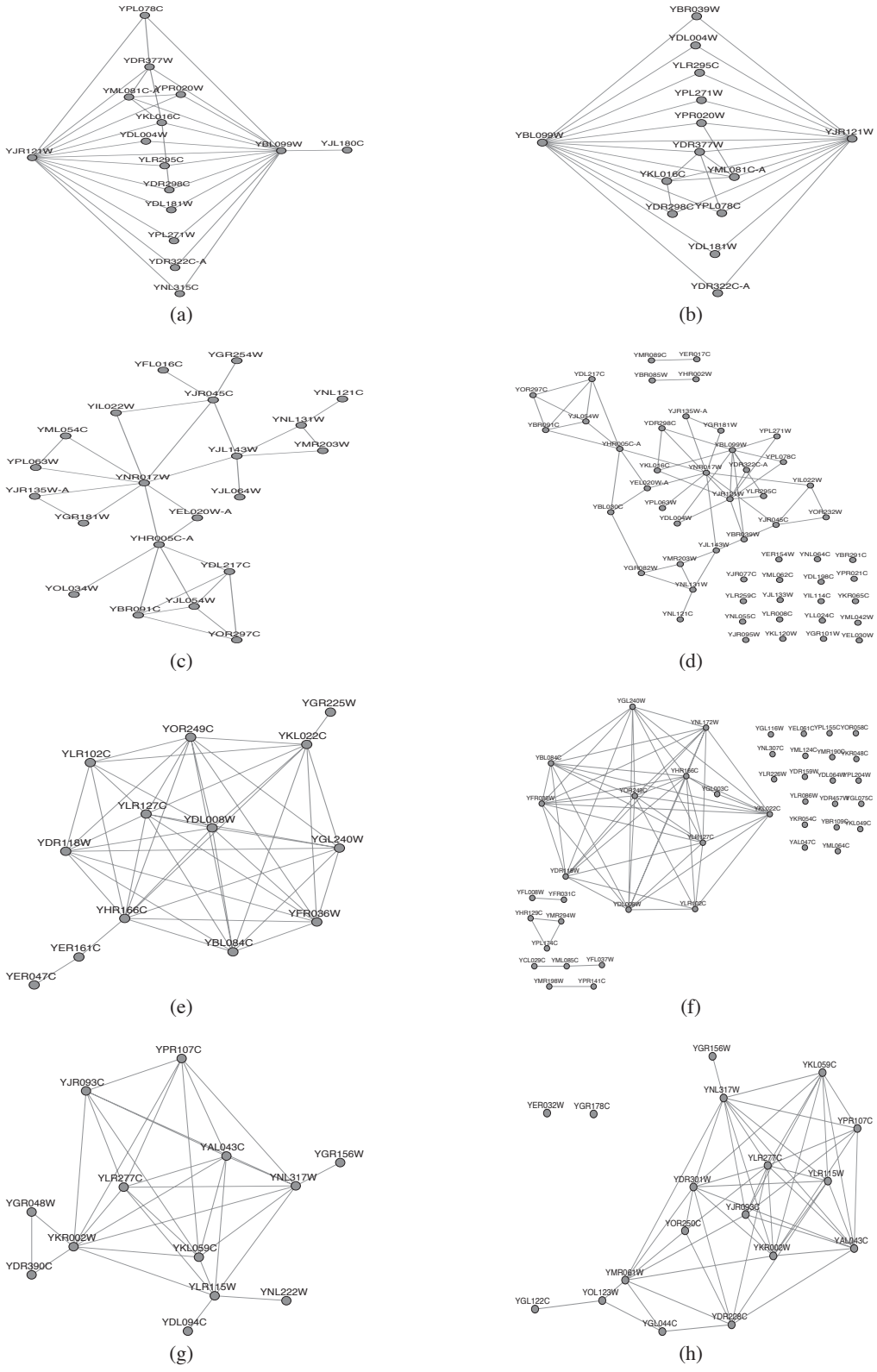


Figure 5: The subgraphs of 4 example clusters identified by STM and their assigned functional categories. The subgraphs of 4 example clusters detected by STM are extracted from yeast PPI network and displayed on the left column in each row. The subgraphs of their associated functional categories, which are assigned from MIPS database for each of these clusters, are also extracted from yeast PPI network and displayed on the right column of their associated clusters.

[8] Hallett, M.B. and Pettit, E.J. . Stochastic events underlie Ca2+ signalling in neutrophils. *J. Theor. Biol.*, 186:1–6, 1997.

[9] Hartuv, E., Shamir, R. A Clustering Algorithm based Graph Connectivity. *Information Processing Letters*, 76:175–181, 2000.

[10] Hishigaki, H., Nakai, K., Ono, T., Tanigami, A. and Takagi, T. Assessment of prediction accuracy of protein function from protein–protein interaction data. *Yeast*, 18:523–531, 2001.

[11] Ho, Y. et al. Systematic identification of protein complexes in Saccharomyces cerevisiae by mass spectrometry. *Nature*, 415:180–183, 2002.

[12] Ito, T. et al. A comprehensive two-hybrid analysis to explore the yeast protein interactome. *PNAS*, 98:4569–4574, 2001.

[13] Johnson, D.B. Efficient algorithms for shortest paths in sparse networks. *J. of the ACM*, 24:1–13, 1977.

[14] Johnson, N.L., Kotz, S. and Balakrishnan, N. *Continuous univariate distributions*. John Wiley & sons, New York, NY, 1994.

[15] Karypis, G., Han, E.-H. and Kumar, V. Chameleon: Hierarchical clustering using dynamic modeling. In *IEEE Computer: Special issue on data analysis and mining*, volume 32, pages 68–75, 1999.

[16] King, A.D., Przulj, N. and Jurisica, I. Protein complex prediction via cost-based clustering. *Bioinformatics*, 20:3013–3020, 2004.

[17] Letovsky, S. and Kasif, S. Predicting protein function from protein/protein interaction data: a probabilistic approach. *Bioinformatics*, 19: i197–i204, 2003.

[18] Lin, C. et al. *Knowledge Discovery in Bioinformatics: Techniques, Methods, and Applications*, chapter Clustering methods in protein-protein interaction network, page . John Wiley & Sons Inc., 2006.

[19] Mewes, H. W. . MIPS: analysis and annotation of proteins from whole genome in 2005. *Nucleic Acid Research*, 34:D169–D172, 2006.

[20] Pereira-Leal, J.B., Enright, A.J. and Ouzounis, C.A. Detection of functional modules from protein interaction networks. *Proteins*, 54:49–57, 2004.

[21] Ramanathan, M. A dispersion model for cellular signal transduction cascades. *Pharm. Res.*, 19:1544–1548, 2002.

[22] Samanta, M.P. and Liang, S. . Predicting protein functions from redundancies in large-scale protein interaction networks. *PNAS*, 100:12579–12583, 2003.

[23] Samanta, M.P. and Liang,S. Redundancies in large-scale protein interaction networks. *Proc. Natl Acad. Sci.*, 100:12579–12583, 2003.

[24] Spirin, V. and Mirny, L.A. . Protein complexes and functional modules in molecular networks. *PNAS*, 100:12123–12128, 2003.

[25] Uetz, P. et al. A comprehensive analysis of protein-protein interactions in Saccharomyces cerevisiae. *Nature*, 403:623–627, 2000.

[26] van Dongen, S. . Technical Report INS-R0010: A cluster algorithm for graphs. *National Research Institute for Mathematics and Computer Science*, 2000.

Protein Folding Trajectories Analysis: Summarization, Event Detection and Consensus Partial Folding Pathway Identification

Hui Yang
Dept. of Comp. Sci. & Eng.
The Ohio State University
2015 Neil Avenue
Columbus, OH 43210, USA
yanghu@cse.ohio-state.edu

Srinivasan Parthasarathy*
Dept. of Comp. Sci. & Eng.
The Ohio State University
2015 Neil Avenue
Columbus, OH 43210, USA
srini@cse.ohio-state.edu

Duygu Ucar
Dept. of Comp. Sci. & Eng.
The Ohio State University
2015 Neil Avenue
Columbus, OH 43210, USA
ucar@cse.ohio-state.edu

ABSTRACT

Molecular dynamics simulations are employed to explain the formation of 3D protein structures, a fundamental yet unsolved problem in computational molecular biology. Effective comparison and representation of the simulation data is a major step in understanding and characterizing the folding process. In this paper, we explore the use of spatio-temporal association patterns to discover meaningful information in protein folding trajectories. We propose an approach that employs the simplicity of contact maps to effectively analyze and utilize information in 3D folding simulations. Our method also allows one to perform cross comparison of multiple trajectories to identify critical events or partial folding pathways common to every trajectory. Our empirical results on the folding trajectories of the protein BBA5 demonstrate the efficacy of the proposed approach.

1. INTRODUCTION

The three dimensional (3D) native structures of proteins have important implications in proteomics. Understanding the structure of a protein enables us to explore the function of the protein, explain substrate and ligand binding, perform realistic drug design and potentially cure diseases caused by misfolding. The protein folding problem is therefore one of the most fundamental yet unsolved problems in computational molecular biology. One major challenge in simulating the protein folding process is its complexity. Snow *et al*. [15] state that performing a Molecular Dynamics (MD) simulation on a mini-protein for just 10 μs would require decades of computation time on a typical CPU. The `Folding@home` distributed computing project [14] recently proposed using worldwide distributed computing to tackle protein folding simulations.

With the increasing number of trajectories produced by distributed computing, there is a need to analyze, understand, and manage the available data. Previously, researchers have examined several summary statistics(e.g. radius of gyration, root mean square deviation (RMSD)) for this purpose. Although summary statistics are commonly used for comparison, they can only capture a biased and limited global property of the conformation. Recently, Russel *et al*. [13] suggested using geometric spanners for mapping a simulation to a more discrete combinatorial representation. They consider using geometric spanners to discover the proximity between different segments of a protein across a range of scales, and track the changes of such proximity over time.

To overcome the difficulties in managing and analyzing the large amount of simulation data, Berrar *et al*. [2] proposed designing a data warehouse system. They embed their warehouse in a grid environment to enable the sharing of the actual simulation data. They also propose implementing a set of data mining algorithms to facilitate commonly needed data analysis tasks.

In this paper, we propose a method to analyze folding trajectories of the mini protein BBA5 produced by the `Folding@home` project. We utilize the spatio-temporal data mining framework that we have developed and described earlier for the purpose of managing and analyzing such data [21]. As mentioned in [21], this framework is designed to analyze spatio-temporal data produced in several scientific domains. In our previous work, we have applied this framework to 8732 proteins taken from the Protein Data Bank to identify structural fingerprints for different classes of proteins [19]. Each protein is associated with a set of objects that are extracted from their contact maps. To effectively capture spatial relationships among objects, we define Spatial Object Association Patterns (SOAP). Furthermore, by associating SOAPs with proteins in different protein classes, we establish the connections between different types of SOAPs and protein classes.

It is apparent that protein folding trajectory data have both spatial and temporal components. Each protein in a MD simulation consists of a number of residues spatially located in the 3D space that move over time. Each frame of the trajectory can be represented as a contact map in 2D, capturing the pair-wise 3D distance of residues. Similar to our previous work [19], we extract non-local bit-patterns from these contact maps. We then use an entropy-based clustering algorithm to cluster such bit-patterns into groups. These bit-patterns are further associated to form spatial object association patterns (SOAPs). By the use of SOAPs, we effectively represent and analyze folding trajectories produced by MD simulations. A major advantage of this representation is its appropriateness for cross-comparison across different simulations. Key benefits of our framework include:

- **Effective, informative and scalable representation of folding simulations**: We represent each frame by a set of SOAPs, where each SOAP in turn characterizes the spatial relationship (or interactions in the folding case) among multiple bit-patterns. SOAPs are not only easily obtainable but also, as

*Contact author.

we will show, are able to capture landmarks along a folding trajectory.

- **Cross-analysis of trajectories to reveal a consensus partial pathway:** By representing each frame as a set of SOAPs, one easily carry out analysis across different trajectories. Such analysis includes detecting critical events and identifying consensus partial folding pathways across trajectories.

2. ANALYSIS OF PROTEIN FOLDING TRAJECTORIES

2.1 Protein Folding Trajectories

Advances in high-performance computing technologies and molecular dynamics have led to successful simulations of folding dynamics for (small) proteins at atomistic level [12]. Such simulations result in a large number of *folding trajectories*, each of which consists of a series of 3D conformations of the protein under simulation. These conformations are usually sampled regularly (e.g., every 200fs) during a simulation. In this paper, we also refer to each conformation as a *folding frame* or simply a *frame*. Furthermore, to represent a protein conformation, we adopt one of the commonly adopted representation schemes, where a conformation is represented as a sequence of α-carbons (C_α) located in 3D space.

As mentioned earlier, we obtained two folding trajectories of the designed mini-protein BBA5 (Protein Data Bank ID) from the `Folding@home` research group at Stanford University[1]. BBA5 is a 23-residue protein that folds at microsecond timescale. The native structure (or fold) of BBA5 shows a β-hairpin involving residues 1-10 and centering about residues 4-5. It also includes an α-helix involving the remaining residues 11-23. By convention, residues are numbered increasingly from the N-terminal to C-terminal of a protein. Figure 1(a) illustrates the native conformation of BBA5. The two folding trajectories, referred to as T_{23} and T_{24} respectively, are of different length. T_{23} consists of a series of 192 conformations (or frames), while T_{24} 150 frames[2]. Each conformation is described at atomistic level in PDB format adopted by the Protein Data Bank programs.

2.2 Comparing Conformations of BBA5 Across Trajectories

Although both trajectories start from the same extended conformation as shown in Figure 1(b), when we examine the visualized frames, they seem to identify two very different folding processes. Figures 1(c) and (d) illustrate the last frame in T_{23} and T_{24} respectively. This seeming difference might be attributed to the stochastic nature of the folding simulation process [12; 16]. However, it is also desirable to characterize the similarities (or dissimilarities) across multiple trajectories.

To compare two trajectories, a key issue that must be addressed is: how can we compare two protein conformations? Several measures have been commonly used to do such comparison, including RMSD (root mean squared distance) [22], contact order [9], and native contacts [4]. However, all these measures are designed to quantify the global topology of a conformation. Furthermore, based on our empirical analysis of these measures, we notice that they are generally too coarse and thus can often be misleading. Even more important, such measures fail to identify similar local structures (or motifs) between conformations. This is especially crucial for small proteins like BBA5. As demonstrated in both experimental and theoretical studies, small proteins often fold hierarchically and begin locally [1]. For instance, it has been shown that BBA5 tends to first form secondary structures such as β-turns and α-helix, then conform to its global topology [16]. Finally, as suggested by Pande [12], both sterics (local motifs) and global topology might play an important role in protein folding. Therefore, to compare conformations of (small) proteins, a more reasonable comparison should consider both local and global structures. Moreover, it should also take the native topology of the protein under study into account.

To meet these requirements, we propose the following approach to compare conformations of BBA5. First, we loosely partition the 23 residues of BBA5 into four fragments: (i) F_1: N-terminal 1-10 β-hairpin; (ii) F_2: C-terminal 11-23 α-helix fragment; (iii)F_3: the first half of F_1 and the second half of F_2; and (iv) F_4: the second half of F_1 and the first half of F_2, i.e., the middle section in the primary sequence. Second, we recognize the secondary structure propensity in each fragment. Two conformations are said to be similar if they demonstrate the same secondary structure propensity in the same fragment. For instance, the conformation pair in Figure 6(a) are similar as residues in F_1,F_2 and F_4 from both conformations indicate a β-turn like local motif. Please note that the orientation of local motifs does not affect the comparison. For instance, in Figure 6(d), we say the two conformations have a similar structure in F_1 fragment, even though the β-turn motifs have different orientations.

To realize the comparison of conformations, two more issues must still be addressed. First, how can we effectively capture and represent local motifs? Second, how can we represent the global topology of a conformation in terms of local motifs? To address the first issue, we leverage the non-local patterns in protein contact maps. For the second, we characterize the spatial arrangement among non-local patterns. Please see Section 3 for details.

2.3 Folding Trajectory Analysis: Objectives

There are two goals we would like to achieve in analyzing the folding trajectories. First, we would like to address the following folding issues for a given trajectory: (1) to detect (or even predict) significant folding events, including the formation of β-turns, α-helices, and native-like conformations; and (2) to recognize the temporal ordering of important folding events in the trajectory. For instance, between the two secondary structures α-helix and β-hairpin in BBA5, which forms earlier? What is ordering of the two events preceding a β-hairpin formation: formation of two extended strands or formation of the turn?

If the first goal concerns individual trajectories, the second goal concerns multiple trajectories. Specifically, we would like to identify a sub-sequence of similar conformations in both trajectories. This sub-sequence of conformations is referred to as the *consensus partial folding pathway*. This is analogous to the Longest Common Sub-sequence (LCS) problem [5], but much more challenging in two important ways. First, we are dealing with time series of 3D protein structures. Second, we are not looking for an exact match between conformations across different trajectories. Instead, we are looking for *similar conformations across trajectories*. We would like to point out that this work is closely related to our previous work on protein structural analysis [18] and our work on mining spatio-temporal data [21].

[1] http://folding.stanford.edu/

[2] Please refer to [12; 16] for details on the simulation model employed to produce such trajectories.

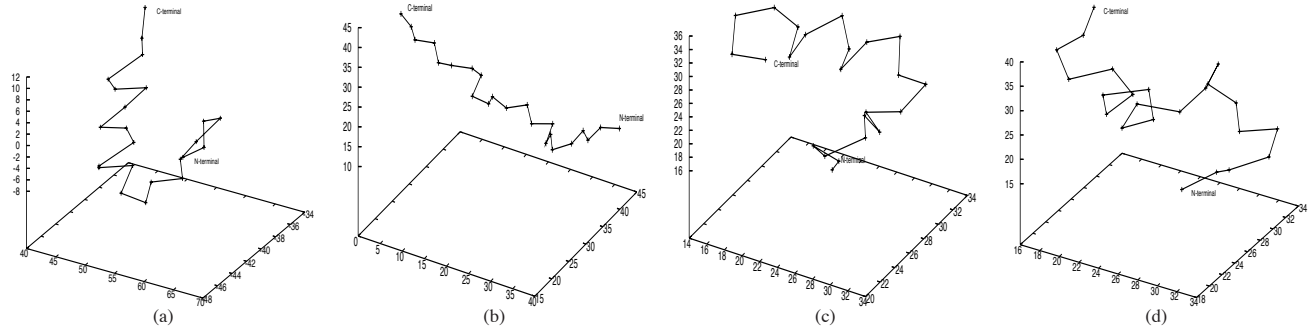


Figure 1: Different conformations of BBA5, where each point corresponds to an α-carbon. (a)The native NMR structure of BBA5 based on data from the SCOP website. (b)The initial conformation of both folding trajectories. (c)The last conformation in the first trajectory. (d)The last conformation in the second trajectory.

3. ALGORITHM

In this section, we describe in detail our approach for analyzing protein folding trajectories. As shown in Figure 2, this analysis consists of three main phases: (I) Data preprocessing, (II) Spatio-temporal object association pattern mining, and (III) Trajectory analysis. We next discuss each phase in further details.

3.1 Data Preprocessing

As in our previous studies on protein structural analysis [18; 19], we represent 3D protein conformations by contact maps. Thus, in this work, we use similar measures to create and process such maps. In order for this algorithm to be self-contained, we briefly go over these preprocessing steps here. We refer to readers to [18] for more detail. In addition, we will also explain the rationale behind several key steps in the context of protein folding.

Contact Map Generation

When generating contact maps, we consider the Euclidean distances between α-carbons (C_α) of each amino acid. Two α-carbons are considered to be in contact if their distance is within 8.5Å. Thus, for a protein of N residues, its *contact map* is an $N \times N$ binary matrix, where the cell at (i, j) is 1 if the i^{th} and j^{th} α-carbons are in contact, 0 otherwise. Since contact maps are symmetric across the diagonal, we only consider the bits below the diagonal. Furthermore, we also ignore the pairs of C_α atoms whose distance in the primary sequence is ≤ 2, as they are sure to be in contact. This step essentially transforms the two BBA5 trajectories into two series of contact maps, with each map of size 23×23.

Identifying Maximally Connected Bit-patterns

Every bit in a contact map has eight neighbor bits. For an edge position, we assume its out-of-boundary positions contain 0. In a contact map, a connected bit-pattern is a collection of bit-1 positions, where for each 1, at least one of its neighbors is 1. Correspondingly, we define a *maximally-connected bit-pattern* (also referred to as a *bit-pattern* in this article) to be a connected pattern p where every neighbor bit not in p is 0. We apply a simple region growth algorithm to identify all *maximally-connected patterns* to every contact map in the two trajectories. A total of 352 maximally-connected bit-patterns were extracted from the two folding trajectories of BBA5.

As discussed in our previous work and elsewhere [7; 8; 10; 17], such bit-patterns can effectively capture the secondary structures of proteins. In the context of protein folding, we have observed that they are powerful enough to represent a wide range of local structural motifs. We can even measure approximately the strength of secondary structure propensity in a conformation based on the bit-patterns. For instance, we have identified bit-patterns that correspond to "premature" α-helices and native-like α-helices respectively. Henceforth, we refer to the 3D structure formed by all the participating residues of a bit-pattern as the *3D motif of the bit-pattern*.

Clustering Bit-patterns into Approximately Equivalent Groups

We apply an entropy-based clustering algorithm to group the bit-patterns into l clusters, where the bit-patterns in a cluster show similar geometric properties (e.g., shape and size). The value of l is determined using an entropy-based measure that quantitatively indicates the quality of a clustering result. (Please refer to [18] for details.) For the BBA5 folding data, the clustering step groups the 352 bit-patterns into 10 clusters (or types).

Intuitively, the 3D motifs of the bit-patterns in a cluster will also have similar 3D geometric properties. This is verified based on our analysis on the BBA5 trajectories. Figure 3 illustrates the representative 3D motifs for 9 of the 10 types of bit-patterns. We omit type 0, as bit-patterns of this type, unlike the others, correspond to a wide variety of 3D motifs.

This demonstrates, to a certain extent, the advantage of using 2D contact maps to analyze 3D protein conformations. Undoubtedly, using contact maps greatly reduces the computational complexity of our algorithm, though at the cost of loss in structural information. More importantly, by exploiting different features in contact maps (bit-patterns in this work), we are able to connect 2D features with features in 3D space. In our case, by identifying 10 types of bit-patterns in contact maps, we indirectly recognize 10 different 3D structural motifs in the folding conformations.

Re-labeling Bit-patterns with The Corresponding Cluster Label

In this step, we re-label all the previously identified bit-patterns with their corresponding cluster label. Let p be a labeled bit-pattern. It can be represented as follows: $p = (trajID, frameID, listC_\alpha, label)$. Here, $trajID$ identifies a folding trajectory, and $frameID$ indicates the frame where p occurs, $listC_\alpha$ consists of all participating α-carbons of p, identified by their position in the primary sequence. Finally, $label$ is the cluster label of p. For BBA5, $label \in \{g_0, g_1, \cdots, g_9\}$, corresponding to the 10 approximately equivalent groups (or types).

I: Data preprocessing
1.1 Generate contact maps for every conformation in the two folding trajectories
1.2 Identify maximally connected bit-patterns in all contact maps
1.3 Cluster bit-patterns into approximately equivalent groups based on geometric properties
1.4 Re-label each bit-pattern with its corresponding cluster label

II: Discovering frequent spatio-temporal object association patterns (SOAPs)
2.1 Discover frequent (minLink=1)-SOAPs of bit-patterns in either folding trajectory

III: Folding trajectory analysis
3.1 Summarize each folding trajectory based on frequent SOAPs
3.2 Detect folding events and recognize the ordering of folding events in a trajectory
3.3 Identify the *consensus partial folding pathway* across trajectories

Figure 2: Main steps to analyze protein folding trajectories

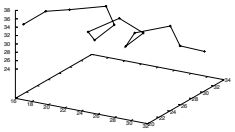
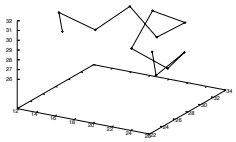
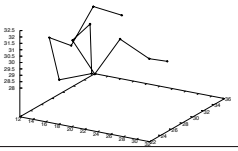
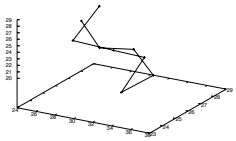
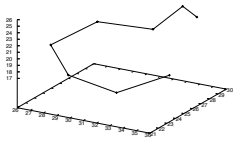
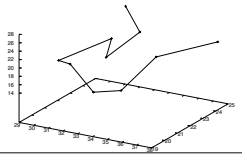
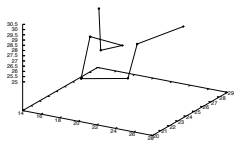
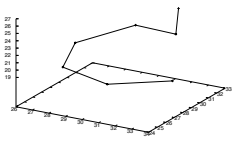
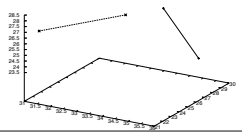
3D Motif	Notation	bit-pattern vs. 3D motif
α-helix like	α	**#4** -111—– 11111—– —-11—- ——11— ——11– ———-11- ———11 ———1- ———1-  **#7** -1———- -11——- –11—– —11—- —111— —1111– —-1111- ——1–1  **#9** -11—– -111—- —11— —111– —-111- —-11– 
β-turn like	β	**#1** –1— –11– -1111- -11— -1—-  **#6** —-1– -11111- -1111–  **#8** —-1— —-11– -111111- —111– —11— —11— —11— 
small α-helix like	a	**#3** (α) -11— -111– —11- ——1  **#2** (β) -1— -11– -111- -1— 
small β-turn like	b	
parallel strands	$\|\|$	**#5** -1– -11- 

Figure 3: Mapping between different types of bit-patterns and 3D motifs in conformations of BBA5 folding trajectories. The bit-patterns and 3D motifs shown here were randomly selected from their respective group for illustration purpose.

3.2 Mining Spatio-temporal Object Association Patterns

The preprocessing steps transform a 3D protein conformation into a set of labeled 2D bit-patterns, that indirectly capture the local 3D structural characteristics of the conformation. For the two BBA5 trajectories, each conformation contains an average of 6 bit-patterns. As BBA5 folds, the dynamics among its 23 residues will be constantly changing until it reaches an equilibrium[3]. This means that two residues previously in contact may become out of contact later. As a result, bit-patterns present in one conformation may be absent in the next. The evolving nature of contacting residues and in turn bit-patterns, is essentially the consequence of a variety of weak interactions among amino acids at different levels. Such weak interactions include hydrogen bonds, electrostatic interactions, van der Waal's packing and hydrophobic interactions [6]. To capture these (potential) interactions, a simple yet effective method is to consider how close two amino acids are located from each other in 3D. We also adopt this method here. Specifically, we consider interactions between local 3D motifs captured by labeled bit-patterns. We denote such interactions as "interactions among bit-patterns". Let p_i and p_j be two bit-patterns in a protein conformation, and $p_i.listC_\alpha$ and $p_j.listC_\alpha$ be the list of α-carbons involved in p_i and p_j, respectively. We define p_i and p_j as *interacting bit-patterns* if at least one pair of α-carbons from $p_i.listC_\alpha$ and $p_j.listC_\alpha$, respectively, are located within a short distance δ. (The value of δ should be greater than the distance that is being used to identify contacting α-carbons when generating contact maps.) In our analysis, we set $\delta = 10$Å.

So far, we have discussed our approach of using bit-patterns in contact maps to characterize local 3D motifs and further represent a protein conformation during folding. We also define the notion of interacting bit-patterns in the folding context. We are ready to present our method of summarizing folding trajectories to fulfill the two objectives described in Section 2.3. The main idea is that we can summarize a folding trajectory by characterizing the evolutionary behavior of interactions among different types of bit-patterns and in turn, the interactions among local 3D motifs.

Definition of (minLink=1) SOAP

As proposed in our previous work [21; 20], such interactions can be modeled and captured by discovering different types of spatial object association patterns (SOAPs). Essentially, SOAPs characterize the specific way that objects, bit-patterns in this case, are interacting with each other at a given time. Among the proposed SOAP types, after a careful evaluation, we empirically select $(minLink = 1)$ SOAPs to model the interacting bit-patterns in the folding process. Let $p = (g_1, g_2, \cdots, g_k)$ be a ($minLink$=1) SOAP of size k, where g_i is one of the 10 types of bit-patterns described above. In the context of folding trajectories, p prescribes that there exists k bit-patterns $b_1, b_2, ..., b_k$ in a conformation, where $b_i.label = g_i$ $(1 \leq i \leq k)$. Furthermore, for each b_i, it interacts with at least one of the remaining $(k-1)$ bit-patterns. Note that the k labels in p are not mutually exclusive. For instance, one can have SOAPs such as $(7\ 9\ 9)$, which involves one type 7 bit-pattern and two type 9 bit-patterns.

We further restrict ourselves to focus only on SOAPs that occur frequently during the folding process (*frequent SOAPs*), since rarely-occurring SOAPS are unlikely to provide reliable insights on the folding process. A SOAP is said to be frequent if it appears in no fewer than $minSupp$ frames in a trajectory. In our studies, we set $minSupp = 5$.

SOAP Episodes

The next step is to capture the evolutionary nature of the folding process. We do this by identifying the evolutionary nature of SOAPs. As mentioned earlier, small proteins like BBA5 often fold hierarchically and begin with local folded structures. As BBA5 folds, new SOAPs can be created and existing one can dissipate. To capture such evolutions, we proposed the concept of ***SOAP episodes***, which provide an effective approach to model the evolution of interactions among spatial objects over time [21]. To reiterate, a SOAP episode E is defined as follows: $E = (p, F_{beg}, F_{end})$, where p is a SOAP composed of one or more bit-patterns, p was created in frame F_{beg} and persisted till frame F_{end}. Note that for a given p, it can be created more than once during protein folding, and thus can have more than one episode.

To discover frequent *(minLink=1)* SOAPs and their episodes in either of the BBA5 trajectories, we apply our SOAP mining algorithm as explained in our previous work [21].

In summary, this mining phase produces the following results: (i) A list of $(minLink = 1)$ SOAPs of bit-patterns that appeared in at least 5 conformations in each folding trajectories; and (ii) A list of episodes, ordered by beginning frame F_{beg}, associated with each of these SOAPs.

3.3 Folding Trajectory Analysis

In this section, we describe our strategy on utilizing SOAPs to summarize a folding trajectory and address the two folding analysis issues described in Section 2.3.

SOAP-based Trajectory Summarization

The previous mining phase discovers a collection of frequent *(minLink=1)* SOAPs and the associated episodes in each trajectory. Therefore, it identifies all the conformations in the trajectories that contain at least one frequent $(minLink = 1)$ SOAPs. For instance, the last conformation in trajectory $T23$ (Figure 1(c)) has two SOAPs of size 2: $(5\ 8)$ (i.e., association of a type 5 and a type 8 bit-pattern) and $(7\ 8)$, and three SOAPs of size 1: (5), (7), and (8), while the last conformation in trajectory $T24$ has three SOAPs: $(7\ 8)$, (7) and (8). This leads to our SOAP-based approach for folding trajectory summarization.

To summarize a folding trajectory, we perform the following three steps. First, for each conformation, we identify all the frequent SOAPs that appear in it and use these SOAPs to represent this conformation. Note that not every conformation contains frequent SOAPs, especially when $minSupp$ is set high. Second, for each SOAP-representable conformation, we carry out the following two tasks on its associated SOAPs. First, for each SOAP, we mark the relative location of each involved bit-pattern in the primary sequence of BBA5. This is done by identifying the segment of BBA5 where the majority of a bit-pattern's α-carbons are located. The segment can be one of the following as described in Section 2.2: F_1, residues $1-10$; F_2, residues $11-23$; F_3, residues 6-17; and F_4: residues 1-5 and 18-23. Let us again take the last conformation in $T24$ as an example. It can be summarized by three SOAPs: $(7\ 8)$, (7) and (8). When we look at the list of α-carbons involved in these bit-patterns, we find out that 7 is mainly located in F_2 and 8 in F_1. Therefore, we mark the three SOAPs as follows: $(8.1\ 7.2)$, (7.2) and (8.1). (We re-arrange the bit-patterns in a SOAP by relative location in BBA5.) This super-imposes BBA5-specific spatial

[3]According to the "folding funnel" theory [11; 3], an equilibrium (or the native folded conformation) has the global minimum energy. However, this might not be the case for simulated folding trajectories.

T_{23}		T_{24}	
frame ID	SOAP	frame ID	SOAP
…	…	…	…
100	(8.1 5.4)	48	(1.1 5.4)
101	(5.2 5.2 1.4)	50	(5.2 5.2 6.4)
103	(6.1 5.2 5.4)	63	(1.1 5.2 5.4)
110	(5.4 1.4)	69	(1.4 5.4)
111	(1.1 3.2)	71	(1.1 2.2)
112	(2.2)	72	(2.2)
…	…	…	…

(a)

T_{23}		T_{24}	
frame ID	SOAP	frame ID	SOAP
…	…	…	…
100	(β.1 $\|$.4)	48	(β.1 $\|$.4)
101	(β.4 $\|$.2 $\|$.2)	50	($\|$.2 $\|$.2 β.4)
103	(β.1 $\|$.2 $\|$.4)	63	(β.1 $\|$.2 $\|$.4)
110	($\|$.4 β.4)	69	($\|$.4 β.4)
111	(β.1 *a*.2)	71	(*b*.2 β.1)
112	(*b*.2)	72	(*b*.2)
…	…	…	…

(b)

Figure 4: SOAP-based trajectory summarization: a segment in either BBA5 folding trajectory. (a) After superimposing the relative location of each bit-pattern and pruning away redundant SOAPs. (b) After further generalizing each bit-pattern by corresponding 3D motif.

information to a SOAP. The next step is to prune away redundant SOAPs after marking each bit-pattern with its relative location in BBA5. A SOAP is redundant if it is embedded in another SOAP. For instance, in the previous example, we can prune away (8.1) and (7.2) as both are embedded in $(7.2\ 8.1)$. After pruning, most conformations in such a small protein can often be represented by a single SOAP. We can even take this summarization a step further, where we replace a bit-pattern with its corresponding 3D motif, as illustrated in Figure 3. For instance, SOAP $(7.2\ 8.1)$ will be transformed into $(\beta.1\ \alpha.2)$. We refer to such SOAPs as *generalized SOAPs*, and the corresponding trajectory as *a generalized trajectory*. Note that in a generalized trajectory, multiple types of bit-patterns can be mapped into a single type of 3D motif. For instance, the α-motif corresponds to three types of bit-patterns 4, 7, and 9(Figure 3). Figure 4 shows a segment in each summarized BBA5 folding trajectory before and after being generalized with 3D motifs.

Detecting Folding Events and Recognizing Ordering Among Events

Once each folding trajectory is summarized into generalized SOAPs, it is fairly straightforward to detect folding events such as the formation of α-helix or β-turn like local structures. This can be done by simply locating the frames that contain the local motif(s) of interest. We can also easily identify native-like conformations, by finding those that contain the generalized SOAP $(\beta.1\ \alpha.2)$. Finally, based on the summarization, one can quickly identify the ordering of folding events in a trajectory. For instance, to check which secondary structure forms more rapidly, α-helix or β-hairpin, one can simply compare the first occurrence of these structures in the summarized trajectory (Figure 4(b)).

Identifying the Consensus Partial Folding Pathway Across Trajectories

To do this, we simply compute the longest common sub-sequence (LCS) in the two summarized trajectories. One can utilize the summarization either before the 3D motif generalization (Figure 4(a)) or after(Figure 4(b)). We use the latter in our analysis. Based on the LCS of generalized SOAPs, we construct the consensus folding pathway by identifying pairs of conformations, one from each trajectory, associated with those generalized SOAPs in the LCS[4]. The resulting consensus pathway is a sequence of conformation pairs of similar 3D structures.

Notice here that the comparison between 3D protein conformations (as described in Section 2.2) is done by using bit-patterns to model local structural motifs, and associations of bit-patterns (SOAPs) to characterize the global structure. It is a hierarchical comparison, which matches the hierarchical folding process of BBA5.

[4]Ambiguity might arise in this process as different conformations can be represented by the same generalized SOAPs. However, this can be easily resolved by taking temporal dimension into account.

4. PRELIMINARY RESULTS

In this section, we report preliminary results on analyzing the two trajectories of protein BBA5. We have described this protein, its trajectories, and the concepts employed in the analysis process in great details earlier. Such information is summarized in Table 1.

4.1 Detecting and Ordering Folding Events

We summarize both folding trajectories with a sequence of SOAPs as illustrated in Figure 4. Coincidently, both summarized trajectories consist of 64 conformations.

Based on these summarized trajectories, we can quickly identify all the conformations where the first α-helix-like or β-turn-like local motifs were formed. For trajectory T_{23}, the first α-helix-like motif was identified in frame 26, and the first β-turn-like local motif was formed in frame 63. For the other trajectory T_{24}, the frames were 29 and 38. This is in accordance with experimental results that α-helices generally fold more rapidly than β-turns. However, since we only consider frequent SOAPs, it is very possible that we might miss the actual first formation of such local motifs. This can easily be overcome by locating the first occurrence of bit-patterns associated with the type of local motif of interest.

For the two events related to β-turn formation, formation of two extended strands and formation of the turn, we found that for both trajectories, the formation of extended strands preceded the formation of the turn.

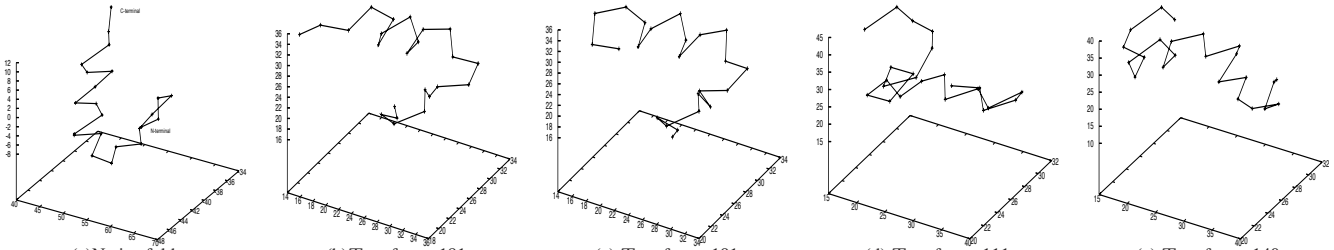
Also, we identify two conformations in each trajectory that show native-like structure. We do this by locating the conformations associated with the generalized SOAP $(\beta.1\ \alpha.2)$. Figure 5 presents the 3D structure of these native-like conformations along with the native conformation of BBA5. One can see that our SOAP-based comparison does well in identifying similar 3D conformations.

4.2 Partial Consensus Folding Pathway Across Trajectory

Based on the generalized trajectory summarization, we identify a consensus partial folding pathway of length 71. In other words, 71 pairs of conformations, one from each trajectory, are considered similar to each other. Furthermore, they evolve from one to the next in the same order. Figure 6 displays four such pairs along this consensus folding pathway. Note that by using bit-patterns, we naturally realize a rotation-invariant comparison. For instance, in Figure 6(d), even though the two β-turns at the N-terminal are of very different orientation, the two conformations are still identified as similar.

Protein	PDB Identifier: BBA5; Primary sequence: 23 residues; Designed protein; Native fold: N-terminal 1-10 β hairpin, C-terminal 11-23 α-helix
Trajectory	Two trajectories: $T23$ and $T24$; $T23$: 192 conformations; $T24$: 150 conformations
Contact map	Based on contacts between α-carbons. Two α-carbons are in contact if their Euclidian distance is ≤ 8.5Å
Bit-patterns	A total of 352 unique maximally connected bit-patterns were identified from all conformations; Average number of bit-patterns per conformation is 6; Bit-patterns are further classified into 10 approximately equivalent types
Interacting bit-patterns	If at least one pair of α-carbons, one from each bit-pattern, is of Euclidian distance$\leq$ 10Å
Frequent SOAPs	A SOAP is frequent if it appears in ≥ 5 conformations; A total of 444 frequent SOAPs identified in trajectory $T23$, and 258 in $T24$
Consensus partial folding pathway	We identified a consensus partial folding pathway across the two trajectories. It is composed of 71 pairs of similar conformations, one from each trajectory

Table 1: A summary of BBA5 folding analysis.



(a)Native fold. (b)T_{23}: frame 181 (c) T_{23}: frame 191 (d) T_{24}: frame 111 (e) T_{24}: frame 140

Figure 5: Native-like conformations in both trajectories based on generalized SOAPs.

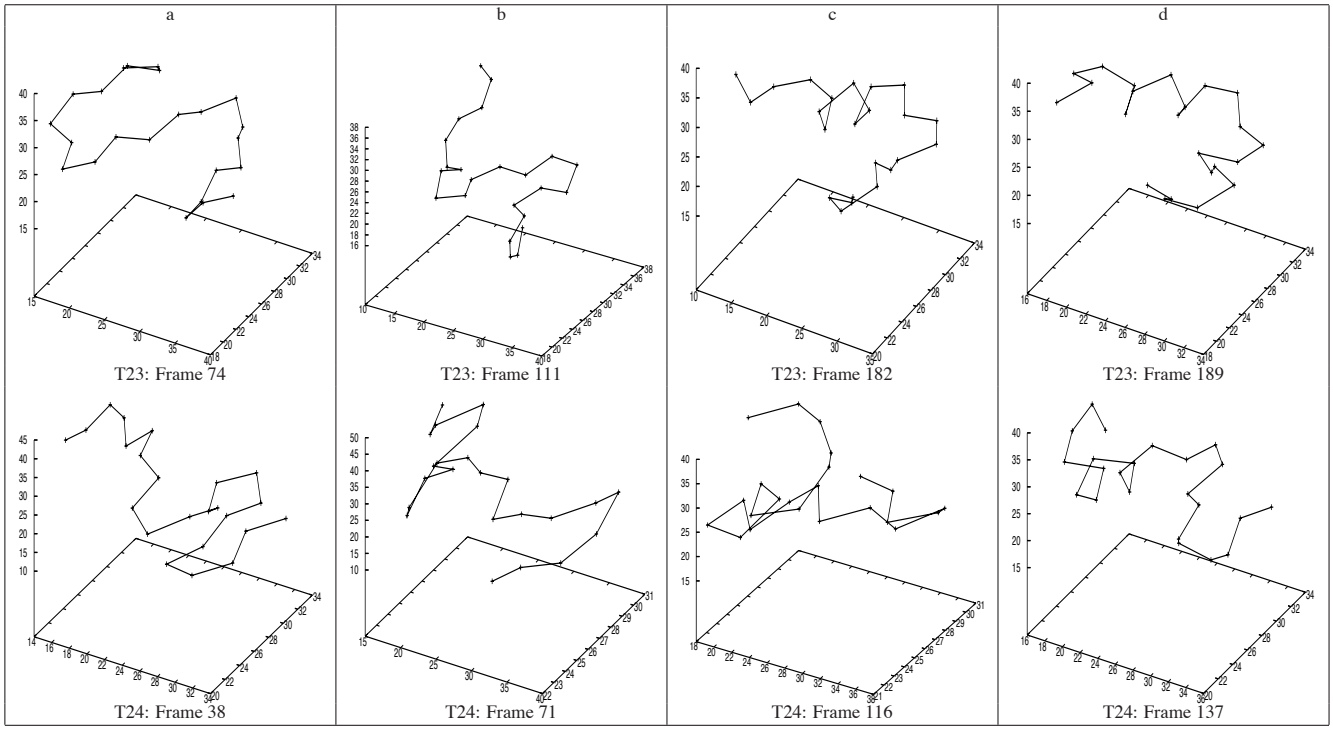


Figure 6: Four pairs of similar conformations on the consensus folding pathway of BBA5.

Currently, we rely on visual tools to justify this pathway. We did attempt to use several measurements that have been used previously to quantify the similarity between 3D protein conformations, but to no avail. These measurements include $RMSD$, contact order, and native contacts. If we identify the pathway based on the best match given by any of the above measurements, where two conformations are a best match if they have the lowest RMSD or have the smallest difference in contact order or native contacts, for instance, we often ended up with a very short consensus pathway (as short as 10 frames). Sometimes, the two conformations along this consensus pathway were visually dissimilar. Moreover, different best-matched measurements rendered very different consensus pathways. We are investigating alternative methods for quantitative validation.

5. CONCLUSIONS AND FUTURE WORK

In this article, we present a novel approach to analyze protein folding trajectories and a case study using the small protein BBA5. We capture a variety of local motifs in the 3D protein conformations by non-local bit-patterns identified in their contact maps. Furthermore, by modeling the interactions or spatial relationships among bit-patterns as SOAPs, we effectively characterize the evolutionary nature of the folding process. We also describe two methods to summarize folding trajectories by super-imposing BBA5-specific information and 3D local structures onto SOAPs. Utilizing the summarized trajectories, we demonstrate that we can detect folding events and the ordering between events, and also identify a consensus folding pathway across trajectories.

We realize that protein folding is a very hard problem. Based on the results of our analysis, we are not in the position to make any general comments on the protein folding problem. However, the approach presented here is meant to be general. It is applicable to any folding trajectories.

There are several issues we plan to address in the future. First, we would like to realize an automatic mapping between bit-patterns and 3D motifs. Second, we will further analyze the consensus folding pathway and validate it through other means. Third, it is well-known that the side chains of a protein play a very crucial role in the folding process. So, we would like to investigate ways to involve side chains in our analysis. We also plan to apply such approach to trajectories of another protein obtained from the `Folding@home` group. Finally, we plan to study whether bit-patterns can be used to index protein folding simulation data.

Acknowledgment: We thank Dr. Yusu Wang for providing the folding simulation data and sharing many constructive and insightful thoughts with us during the project.

6. REFERENCES

[1] Robert L. Baldwin and George D. Rose. Is protein folding hierarchic? i. local structure and peptide folding. *Trends in Biochemical Sciences*, 24(1):26–33, January 1999.

[2] D Berrar, F Stahl, C Silva, JR Rodrigues, RM Brito, and W Dubitzky. Towards data warehousing and mining of protein unfolding simulation data. *Journal of clinical monitoring and computing*, 19(4-5):307–17, October 2005.

[3] J. D. Bryngelson, J. N. Onuchic, N. D. Socci, and P. G. Wolynes. Funnels, pathways, and the energy landscape of protein folding - a systhesis. *Proteins: Struct. Funct. Genet.*, 21(3):167–195, 1995.

[4] Makarov DE, Keller CA, Plaxco KW, and Metiu H. How the folding rate constant of simple, single-domain proteins depends on the number of native contacts. *Proc. Natl. Acad. Sci. USA*, 99::3535–39, 2002.

[5] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman. ISBN 0716710455. A421: SR10, 1979.

[6] Samuel R. Griffiths-Jones. *Peptide models for protein beta-sheets*. PhD thesis, University of Nottingham, 2000.

[7] J. Hu, X. Shen, Y. Shao, C. Bystroff, and M.J. Zaki. Mining non-local structural motifs in proteins. In *BIOKDD 2002*, Edmonton, Canada, 2002.

[8] Jingjing Hu, Xiaolan Shen, Yu Shao, Chris Bystroff, and Mohammed J. Zaki. Mining protein contact maps.

[9] Plaxco KW, Simons KT, and Baker D. Contact order, transition state placement and the refolding rates of single domain proteins. *J. Mol. Biol.*, 277:985–994, 1998.

[10] Giuseppe Lancia, Robert Carr, Brian Walenz, and Sorin Istrail. 101 optimal pdb structure alignments: a branch-and-cut algorithm for the maximum contact map overlap problem. In *Proceedings of the fifth annual international conference on Computational biology*, pages 193–202. ACM Press, 2001.

[11] P. E. Leopold, M. M. J. N, and Onuchic. Protein folding funnels - a kinetic approach to the sequence structure relationship. *Proceedings of National Academy Science, USA*, 89(18):8721–8725, 1992.

[12] Vijay S. Pande. Meeting halfway on the bridge between protein folding theory and experiment. *Proceedings of National Academy of Sciences*, 100(7):3555–3556, April 2003.

[13] D Russel and L Guibas. Exploring protein folding trajectories using geometric spanners. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, pages 40–51, 2005.

[14] M Shirts and V S Pande. Screen savers of the world unite. *Science*, 290:1903–1904, 2000.

[15] C D Snow, H Nguyen, V S Pande, and M Gruebele. Absolute comparison of simulated and experimental protein-folding dynamics. *Nature*, 420:102–106, 2002.

[16] Christopher D. Snow, Houbi Nguyen, Vijay S. Pande1, and Martin Gruebele. Absolute comparison of simulated and experimental protein-folding dynamics. *Nature*, 420:102–106, November 2002.

[17] Michele Vendruscolo and Eytan Domany. Efficient dynamics in the space of contact maps. *Folding & Design*, 3(5):329–336, 1998.

[18] H. Yang, K. Marsolo, S. Parthasarathy, and S. Mehta. Discovering spatial relationships between approximately equivalent patterns. In *BIOKDD 2004*, August 2004.

[19] H. Yang, S. Mehta, and S. Parthasarathy. Mining spatial object patterns in scientific data. In *Proceedings of the 19th Internationl Joint Conference of Artificial Intelligence (IJCAI)*, 2005.

[20] H. Yang, S. Parthasarathy, and S. Mehta. Towards association based spatio-temporal reasoning. In *Proceedings of the 19th IJCAI Workshop on Spatio-temporal Reasoning*, 2005.

[21] Hui Yang, Srinivasan Parthasarathy, and Sameep Mehta. A generalized framework for mining spatio-temporal patterns in scientific data. In *KDD '05: Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 716–721, New York, NY, USA, 2005. ACM Press.

[22] Bojan Zagrovic, Christopher D. Snow, Siraj Khaliq, Michael R. Shirts, and Vijay S. Pande. Native-like mean structure in the unfolded ensemble of small proteins. *J. Mol. Biol.*, 323:153–164, 2002.

Invited Talk

David Roos
Merriam Professor of Biology
Director, Penn Genomics Institute
University of Pennsylvania
Philadelphia, PA 19104 USA
droos@sas.upenn.edu

Automatic Layout and Visualization of Biclusters

Gregory A. Grothaus*
Google Inc.
1600 Amphitheater Parkway
Mountain View CA 94043
ggrothau@gmail.com

Adeel Mufti
Dept. of Computer Science
660 McBryde Hall
Virginia Polytechnic Institute
and State University
Blacksburg VA 20460
amufti@vt.edu

T. M. Murali
Dept. of Computer Science
660 McBryde Hall
Virginia Polytechnic Institute
and State University
Blacksburg VA 20460
murali@cs.vt.edu

ABSTRACT

Biclustering has emerged as a powerful algorithmic tool for analyzing measurements of gene expression. A number of different methods have emerged for computing biclusters in gene expression data. Many of these algorithms may output a very large number of biclusters with varying degrees of overlap. There are no systematic algorithms that create a two-dimensional layout of the computed biclusters and display overlaps between them.

We develop a novel algorithm for laying out biclusters in a two-dimensional matrix whose rows (respectively, columns) are rows (respectively, columns) of the original dataset. We display each bicluster as a contiguous submatrix in the layout. We allow the layout to have repeated rows and/or columns from the original matrix as required, but we seek a layout of the smallest size. We also develop a web-based search interface for the user to query the layout for genes and samples of interest. We demonstrate the usefulness of our approach on gene expression data for two types of leukemia and on protein-DNA binding data for two growth conditions. The software implementing the layout algorithm is available at http://bioinformatics.cs.vt.edu/~murali/papers/bivoc.

1. INTRODUCTION

Measurement of gene expression using DNA microarrays [13; 31] have revolutionized biological and medical research. Since gene expression plays an important role in cell differentiation, development, and pathological behavior, computational analysis of DNA microarray data has the potential to assign functions to newly-discovered genes, unravel the structure of biological pathways, and assist in the development of new medicines. Biclustering has emerged as a powerful algorithmic tool for analyzing gene expression data. A bicluster in a gene expression data set is a subset of genes and a subset of conditions with the property that the selected genes are co-expressed in the selected conditions; these genes may not have any coherent patterns of expression in the other conditions in the data set. Biclusters have a number of advantages over clusters computed by more traditional algorithms such as k-means and hierarchical clustering [11]. Since a bicluster includes only a subset of genes and samples, it models condition-specific patterns of co-expression. Traditional clusters may miss such patterns since they operate in the space spanned by all the conditions. Further, many biclustering algorithms allow a gene or a sample to participate in multiple biclusters, reflecting the possibility that a gene product may be a member of multiple pathways.

A number of different methods have emerged for computing biclusters in gene expression data [6; 5; 9; 14; 17; 20; 22; 28; 32; 33; 34; 35; 36]; two papers survey these techniques [26; 37]. These algorithms use a number of different strategies to compute biclusters such as exhaustive enumeration [8; 24; 36], iterated improvement [5; 9], repeated random sampling [28], and expectation maximization [32]. An issue all these algorithms deal with is trying to avoid outputting two or more biclusters with nearly the same set of samples and/or genes. A common approach is to remove a bicluster from the output if it shares a large fraction of genes and/or samples (based on a user-defined threshold) with an already computed bicluster. In spite of these measures, biclustering algorithms may compute tens, hundreds, or even thousands of biclusters with varying degrees of overlap.

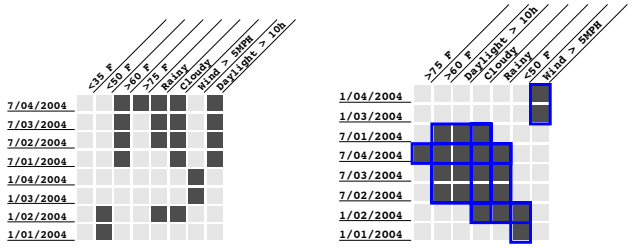
Organising, manipulating, and querying the potentially large number of biclusters computed by these algorithms is a data mining task in itself, which has not been adequately addressed. In this paper, we develop a novel algorithm for laying out biclusters in a manner that visually reveals overlaps between them. We lay out the biclusters in a two-dimensional matrix whose rows (respectively, columns) are rows (respectively, columns) of the original dataset. We display each bicluster as a contiguous submatrix in the layout. We allow the layout to have repeated rows and/or columns from the original matrix, but we seek a layout of the smallest size. In addition, we develop a web-based search interface that allows the user to query the results for genes and samples of interest and visualise the layout of the biclusters that match the search criteria.

The layout algorithm is general enough to be applied to biclusters computed in real-valued, binary, or categorical data. For instance, the combination of biclustering algorithms and our layout algorithm can be used to analyze measurements of the concentrations of other types of molecules, including proteins and metabolites. We demonstrate our approach on two types of data. First, we compute layouts for biclusters extracted from leukemia microarray data by the xMotif biclustering algorithm [16; 28]. Second, we analyze protein-DNA binding data in *S. cerevisiae* and demonstrate how

*This author performed the research at the Virginia Polytechnic Institute and State University.

biclustering in combination with the layout algorithm can visually demonstrate differences in the transcriptional regulatory network that is activated in different growth conditions.

Figure 1 displays a layout computed by our algorithm on a toy binary matrix. Figure 1(a) displays a dataset in which rows represent dates and columns represent weather conditions in Blacksburg. A cell has a one (the cell is drawn shaded) if the weather condition corresponding to the cell's column (e.g., "Rainy" or "$>$ 75° F") is true on the date corresponding to the cell's row. In this dataset, we define a bicluster to be an itemset, i.e., a subset of rows and a subset of columns with the property that the submatrix defined by these rows and columns only contains ones. We computed all the closed biclusters in this binary matrix, i.e., biclusters with the property that every row (respectively, every column) not in the bicluster contains a zero in at least one column (respectively, one row) in the bicluster. Figure 1(b) displays the layout computed by our algorithm of the seven biclusters in this dataset. .



(a) Weather conditions in Blacksburg.

(b) Layout of the biclusters in the weather data.

Figure 1: An example of a bicluster layout for weather data in Blacksburg, VA.

The bicluster layout problem, which we formally define in Section 3.1, is very similar to the hypergraph superstring problem studied by Batzoglou and Istrail study in the context of physical mapping of genomes. Batzoglou and Istrail prove that the hypergraph superstring problem is MAX-SNP Hard, i.e., it is computationally intractable to obtain a bicluster layout whose size is smaller than a constant factor of the optimal size. In this work, we present a heuristic that minimizes the size of the layout well in practice. In the special case when there is a solution involving no repeated rows or columns, the algorithm computes the layout of smallest size. Our algorithm runs in $O(mn^2 + n^2 \log n)$ where n is the number of biclusters and m is the number of rows and columns in all the biclusters; the running time of the algorithm is independent of the size of the original dataset. We lay out the rows and columns of the biclusters independently. Our algorithm to lay out the rows is similar to a bottom-up hierarchical clustering of the rows of the biclusters. At each stage, we merge two biclusters if the submatrix induced by them in the original matrix has the "consecutive ones property" (see Section 3.2). Finally we generate the two-dimensional layout by combining the row and column layouts.

2. RELATED WORK

A binary matrix has the *Consecutive Ones Property* (COP) for rows if its columns can be permuted such that all the ones in each row are consecutive [7]. See Figure 2 for an example of a matrix with the COP. Determining whether a matrix has the COP and computing the permutation of the columns that proves this property has applications in a number of areas including testing for graph planarity [7] and recognizing interval graphs [7; 18]. Booth and Leuker [7] describe a data structure called the PQ tree which they use to represent all legal permutations of column orderings in a matrix with the COP property. They prove that the PQ tree and the correct column permutation can be computed in time linear in the number of ones in the matrix.

$$\begin{pmatrix} \mathbf{1} & \mathbf{0} & 1 & 1 & 0 \\ \mathbf{0} & \mathbf{0} & 1 & 1 & 1 \\ \mathbf{1} & \mathbf{0} & 1 & 0 & 0 \\ \mathbf{1} & \mathbf{1} & 1 & 1 & 1 \\ \mathbf{0} & \mathbf{0} & 0 & 1 & 1 \end{pmatrix}$$

(a) A matrix that has the COP with the first two columns highlighted.

$$\begin{pmatrix} \mathbf{0} & \mathbf{1} & 1 & 1 & 0 \\ \mathbf{0} & \mathbf{0} & 1 & 1 & 1 \\ \mathbf{0} & \mathbf{1} & 1 & 0 & 0 \\ \mathbf{1} & \mathbf{1} & 1 & 1 & 1 \\ \mathbf{0} & \mathbf{0} & 0 & 1 & 1 \end{pmatrix}$$

(b) Swapping the first two columns of the matrix demonstrates that the matrix has the COP.

Figure 2: An illustration of the COP.

Researchers have studied a number of generalizations of the COP problem; however, most of these generalizations are NP-complete or NP-Hard. For example, seeking the column ordering for a non-COP matrix that minimizes the number of gaps between the ones in each row can be reduced to the traveling salesman problem [12]. An important generalization studied in bioinformatics is one where we are allowed to repeat as well as rearrange columns in order to ensure that the consecutive ones of every original row occur in at least one contiguous set of columns in that row. As mentioned earlier, in their study of physical mapping of chromosomes, Batzoglou and Istrail prove that this problem is MAXSNP-Hard [3]. The most common application of this generalization of the COP is physical mapping of chromosomes with probes. We can represent physical mapping data as a binary matrix where the rows represent clones (short overlapping sections of a chromosome), the columns represent DNA probes, and a cell in the matrix has a one if the corresponding probe hybridizes to the corresponding clone. Constructing a physical map of the chromosome is equivalent to finding an ordering of the probes such that all the probes matching a clone appear consecutively and the total length of the ordering is as small as possible.

Algorithms for constructing physical maps from hybridization data typically exploit the Lander-Waterman model [21], which assumes that clones are distributed uniformly across the chromosome and that probes are distributed according to independent Poisson processes. Some algorithms make additional domain-specific assumptions [3; 12; 19; 25; 27]. For instance, Batzoglou and Istrail compute an ordering whose length is at most twice the length of the optimal ordering under the requirement that each clone must contain a probe that does not hybridize to any other clone. None of these algorithms are applicable to our problem since the biclusters we want to lay out may not have the required properties.

3. ALGORITHM

We present our approach in four stages. First, we define some useful notation. Second, we introduce the PQ-tree, a data structure that is fundamental to our approach. Third, we present our layout algorithm. Finally, we discuss its implementation and the web interface.

3.1 Definitions

We denote the input matrix by D and use R and C to denote the set of rows and columns of D, respectively. Given subsets $R' \subseteq R$ and $C' \subseteq C$, we define a *bicluster* $B(R', C')$ to be the sub-matrix of D spanned by the rows in R' and the columns in C'.[1] A *layout* $\mathcal{L}(\mathcal{R}, \mathcal{C})$ of the matrix D is a two-dimensional matrix specified as follows:

1. $\mathcal{R}$ is the ordered list of rows of $\mathcal{L}$ with the property that each element of $\mathcal{R}$ is an element of R; a row in R can appear multiple times in $\mathcal{R}$.

2. $\mathcal{C}$ is the ordered list of of columns of $\mathcal{L}$ with the property that each element of $\mathcal{C}$ is an element of C; a column in C can appear multiple times in $\mathcal{L}$.

3. $\mathcal{L}_{ij}$, the element in the ith row of $\mathcal{R}$ and the jth column of $\mathcal{C}$ is equal to $D_{i'j'}$, where i' is the row of D corresponding to the the ith row of $\mathcal{L}$ and j' is the column of D corresponding to the jth column of $\mathcal{R}$.

The *size* of $\mathcal{L}$ is $|\mathcal{R}||\mathcal{C}|$. It is appropriate to consider $\mathcal{L}$ to be a layout of D since $\mathcal{L}$ specifies an order for the rows and columns of D. In the example in Figure 1(b), the layout does not contain any repeated rows or columns. The layout does not contain the column titled "< 35 F" that is in the original matrix either.

A bicluster $B(R', C')$ is *contiguous* in a layout $\mathcal{L}(\mathcal{R}, \mathcal{C})$ if and only if the elements of R' (respectively, C') appear consecutively at least once in $\mathcal{R}$ (respectively, $\mathcal{C}$). We say that the layout $\mathcal{L}(\mathcal{R}, \mathcal{C})$ is *valid* with respect to a set of biclusters S if every bicluster $B \in S$ is contiguous in $\mathcal{L}(\mathcal{R}, \mathcal{C})$. For example, the layout in Figure 1(b) is valid with respect to the bicluster ({7/04/2004, 7/03/2004, 7/02/2004 }, { > 60F, Daylight > 10h, Cloudy, Rainy }) since the bicluster spans rows four to six and columns two to five in the layout. We now formally define the *bicluster layout* problem: Given a matrix D and a set S of biclusters in D, find a layout $\mathcal{L}$ of D such that $\mathcal{L}$ is valid with respect S and $\mathcal{L}$ has the smallest size among all valid layouts of D.

3.2 PQ Trees

Booth and Leuker [7] developed a data structure called the PQ tree, which they used to compute a column ordering that proves that that a binary matrix M has the COP. To define the PQ tree, it is convenient to reformulate the COP problem as follows: Let U be the set of columns of M. Let r be the number of rows in M. For each $i, 1 \leq i \leq r$, define the set S_i to be the set of columns in U that have a one in row i. We seek a permutation of the elements of U that satisfies r *restrictions*, where restriction $i, 1 \leq i \leq r$ requires that the elements of S_i be consecutive in the permutation.

[1]This simple definition is sufficient for this paper. An algorithm that computes biclusters in gene expression data is likely to use a more complex definition relevant to the patterns to be mined.

A PQ tree can represent all legal permutations of U that satisfy the restrictions $\{S_i, 1 \leq i \leq r\}$. Each leaf of the PQ tree corresponds to a column in U. The PQ tree contains two types of internal nodes: P-nodes and Q-nodes. The children of a P-node can be permuted in any way while still satisfying the restrictions. A valid permutation of the children of a Q-node is either the order in which they appear in the PQ tree or the reversal of this order. A PQ tree supports the REDUCE operation. This operation inserts a restriction S into a PQ tree T. The operation modifies T such that T satisfies S in addition to all the previous restrictions inserted into T. The REDUCE operation fails if there are no legal permutations of U that can satisfy S and the previously inserted restrictions. The REDUCE operation takes time linear in $|S|$. Figure 3 displays a PQ tree on four elements $\{a, b, c, d\}$ after two REDUCE operations: REDUCE(T,$\{a, c\}$) and REDUCE(T,$\{b, c\}$). Inserting the restriction $\{c, d\}$ into the tree next will result in a failed REDUCE operation.

To solve the COP problem, start with an empty PQ tree T. For each $i, 1 \leq i \leq r$, invoke the operation REDUCE(T, S_i). To obtain an ordering that satisfies the restrictions, perform a breadth-first traversal of T starting at the root. At each internal node of T, visit the children of the node in an order specified by the type of the node. At a leaf node of T, append the column corresponding to the leaf to the required ordering.

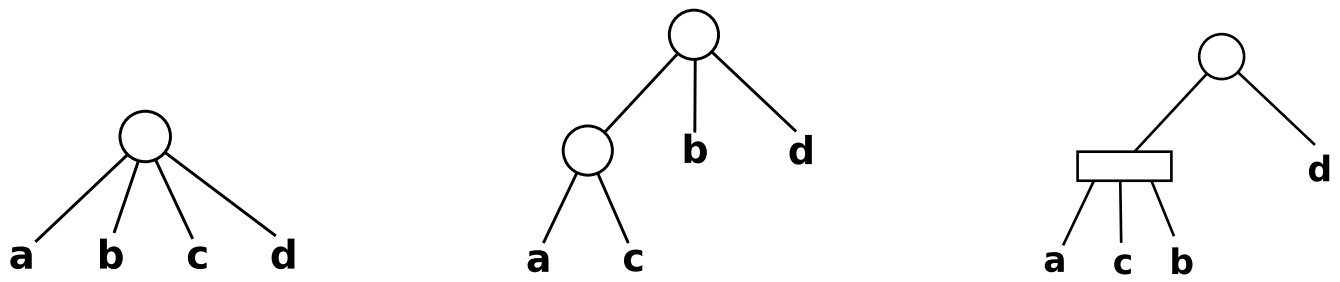
3.3 The Bicluster Layout Algorithm

We are now ready to describe our algorithm for the bicluster layout problem. To minimize the size of $\mathcal{L}$, we can minimize the length of $\mathcal{R}$ and the length of $\mathcal{C}$ independently. Therefore, we construct the layout $\mathcal{L}$ by determining $\mathcal{R}$ and $\mathcal{C}$ independently. In the rest of this section, we describe the algorithm to construct $\mathcal{C}$, the ordered list of the columns in the layout $\mathcal{L}$. We can compute $\mathcal{R}$, the ordered list of rows in the layout, analogously.

We describe the algorithm in two stages. We first transform the problem of constructing $\mathcal{C}$ to a generalization of the COP problem. We then present an algorithm to solve this transformed problem. This transformation allows us to describe our algorithm in terms of operations on PQ trees. The PQ tree cannot solve this generalization directly since the matrix we construct may not have the COP.

We start by constructing a new binary matrix M that represents the columns of the biclusters in S. Each column on M corresponds to a column of the input matrix D. M contains one row for each bicluster in S; thus, M has n rows. The entry M_{ij} is 1 if the ith bicluster in S contains the column j in D; otherwise, M_{ij} is 0. We can now reformulate the problem of constructing $\mathcal{C}$ as follows: find the shortest linear ordering $\mathcal{C}$ of the columns of M such that $\mathcal{C}$ can contain repeated columns of M and for every row of M, the columns containing the ones in that row appear consecutively at least once in $\mathcal{C}$.

Before describing the algorithm, we define some more notation. The leaves of each PQ tree constructed by the algorithm correspond to a subset of the columns of M. We use C_T to denote the set of columns in a PQ tree T. Given two PQ trees T and T', let $\sigma(T, T')$ denote the set similarity $\frac{|C_T \cap C_{T'}|}{|C_T \cup C_{T'}|}$ between the columns in T and T'. Our algorithm executes the following steps:



(a) Initial PQ tree T formed from set $\{a, b, c, d\}$.

(b) The PQ tree T after the REDUCE(T,$\{a, c\}$) operation, requiring that a and c be consecutive.

(c) The PQ tree T after the REDUCE(T,$\{b, c\}$) operation, requiring that b and c be consecutive.

Figure 3: An example of a PQ tree. Circles represent P nodes and rectangles represent Q nodes. Valid permutations represented by the tree in Figure 3(c) are the sequences $acbd, bcad, dacb$, and $dbca$.

1. For each row i of M, $1 \leq i \leq n$, construct a PQ tree T_i consisting of a single P-node, whose children are the columns in M that contain ones in row i of M. Let $\mathcal{T}$ be the set of these n PQ trees.
2. For every pair $1 \leq i < j \leq n$, compute the set similarity $\sigma(T_i, T_j)$.
3. Compute Σ, the list of values in $\{\sigma(T_i, T_J), 1 \leq i < j \leq n\}$ sorted in descending order.
4. Repeat the following steps until Σ is empty:
   (a) Remove the largest element from Σ. Let T and T' be the PQ trees in $\mathcal{T}$ with this similarity value.
   (b) Set $T'' = T$.
   (c) For each restriction r inserted into T', invoke the operation REDUCE(T'', r). If any reduce operation fails, go to Step 4a.
   (d) Delete T and T' from $\mathcal{T}$.
   (e) For each tree $U \in \mathcal{T}$, insert $\sigma(U, T'')$ into Σ.
   (f) Insert T'' into $\mathcal{T}$.
5. For each PQ tree T in $\mathcal{T}$, traverse T to compute a valid permutation of the columns in C_T.
6. Output the column layout formed by concatenating (in any order) the permutations computed in Step 5.

The algorithm starts by storing each row of M (recall that each row of M corresponds to a bicluster) in a separate PQ tree in the set $\mathcal{T}$ (Step 1). Next, the algorithm performs a series of REDUCE operations to hierarchically cluster the rows of M. Inductively, each PQ tree in $\mathcal{T}$ corresponds to a set of rows of M with the property that the submatrix of M defined by these rows has the COP. To decide which two sets of rows to merge next, in Step 4a, the algorithm picks the two PQ trees T and T' in $\mathcal{T}$ that are the most similar and attempts to merge them. To effect the merger, the algorithm adds the restrictions added to one of these PQ trees to the other PQ tree (Step 4c). If this step succeeds, the algorithm deletes T and T' from $\mathcal{T}$, inserts the similarities between the new PQ tree T'' and each of the remaining PQ trees in $\mathcal{T}$ into Σ, and inserts T'' into $\mathcal{T}$ (Steps 4d–4f).

In Step 4c, the failure of a REDUCE operation means that the restrictions in T are not compatible with the restrictions imposed by T'. Hence, the submatrix of M induced by the union of rows in T and in T' does not have the COP. An example of such a situation is when T corresponds to the tree in Figure 3(c) and T' contains the restriction $\{c, d\}$. In this case, the algorithm aborts the merger of T and T' and moves on to the next most similar pair of PQ trees. Due to such conflicts, $\mathcal{T}$ may contain more than one PQ tree when the algorithm completes. Finally, generating the required layout is a simple matter of traversing each PQ tree in $\mathcal{T}$ (Step 5) as described in Section 3.2 and concatenating the resulting permutations into a single order (Step 6). A column of M appears as many times in this order as there are PQ trees in $\mathcal{T}$ that includes this column.

We now analyze the running time of the algorithm. Let m be the number of ones in the matrix M. In Step 1, computing the PQ trees takes $O(m)$ time. Computing the similarity between a pair of PQ trees takes $O(c)$ time, where c is the number of columns of M. In Step 2 and 3, computing and sorting the $O(n^2)$ similarity values takes $O(cn^2 + n^2 \log n)$ time. We execute Step 4 $O(n^2)$ times. The running time of each iteration is proportional to the size of the new PQ tree constructed. A naive upper bound on this size is m, the total number of columns in all the biclusters. Hence, the total running time of Step 4 is $O(mn^2)$. Finally, traversing all the PQ trees in $\mathcal{T}$ and concatenating the permutations takes $O(m)$ time. Keeping in mind that $c \leq m$, the total running time of the algorithm is $O(mn^2 + n^2 \log n)$. The space occupied by the algorithm is $O(m + n^2)$, with $O(m)$ space taken to store all the biclusters and the PQ trees and $O(n^2)$ required for Σ, the sorted list of similarities.

3.4 Implementation and Web Interface

We implemented the layout algorithm in C++ and tested it on a 2.8GHz Pentium computer running the Fedora Core 3 operating system. Our software contains two executable programs. The first executable, `layout`, implements the layout algorithm. It takes a text file describing the biclusters as input and outputs the layout as a text file list of rows and columns. The second executable, `drawlayout`, uses this text file and the original data set as input and produces an image corresponding to the layout.

If the input data contains a large number of biclusters, the layout may contain too many rows and/or columns for the user to navigate with ease. To alleviate this problem, we have also developed a simple web-based interface that allows the user to upload a file containing computed biclusters and a file containing the original data, and search the biclusters with the names of rows and columns. The interface invokes `layout` and `drawlayout` on the biclusters that contain the query rows/columns and highlights the matching biclusters, rows, and columns in the resulting layout. The interface allows the user to specify whether the data is real-valued or binary, whether the layout should contain only the matching biclusters, and whether the query should be a conjunction or disjunction of the search terms.

4. EXPERIMENTAL RESULTS

4.1 Synthetic Data

We created synthetic datasets with different numbers of rows and columns. For each dataset, we generated biclusters by sampling subsets of rows and columns. For this experiment, we randomly generate the number of rows and columns and identifiers for the rows and columns; we did not need to generate values for the cells of the matrices. For each set of biclusters, we recorded the time required to run our layout algorithm and the number of rows and columns in the computed layout. For each layout, we estimated the *layout efficiency* of the layout as the ratio of the size of the layout to the size of the dataset. Lower values of efficiency are better than higher values, since they indicate that the algorithm is able to exploit overlaps between biclusters. For each choice of number of rows in the dataset, number of columns in the dataset, and number of biclusters, we averaged the results for 100 runs. Tables 1 and 2 display our results. Efficiency values may be less than one, e.g., when some rows or columns in the dataset do not belong to any bicluster.

Table 1: Execution times (in seconds) for the layout algorithm on synthetic matrices.

#biclusters	#rows + #columns in the dataset				
	10	30	50	70	90
20	0.168	0.328	0.462	0.52	0.532
40	1.23	2.514	3.046	3.574	4.008
60	4.074	7.992	11.238	11.71	12.81
80	9.484	19.586	25.546	29.652	29.446
100	17.982	37.966	48.418	50.916	56.112

Table 2: Efficiency values of the layout algorithm on synthetic matrices.

# biclusters	#rows + #columns in the dataset				
	10	30	50	70	90
20	0.184	0.842	1.316	1.254	1.428
40	0.304	1.16	1.632	2.04	2.074
60	0.398	1.496	2.262	2.26	2.508
80	0.512	1.65	2.358	2.726	2.698
100	0.48	1.808	2.582	2.686	2.996

4.2 Transcriptional Regulation in *S. cerevisiae*

To demonstrate the ability of our visualization algorithm to highlight differences between biclusters in similar datasets, we analyzed datasets of transcriptional regulation in two experimental conditions in *S. cerevisiae* [2; 23]. Each dataset is a binary matrix whose columns represent transcription factors and whose rows represent genes in *S. cerevisiae*. A matrix entry contains a one if a ChIP-on-chip experiment indicates that the transcription factor binds to the promoter of the gene. An important problem that arises in the analysis of this data is determining if a set of genes are collectively regulated by a set of transcription factors and whether this combinatorial regulation changes when the cell is exposed to stress. Although ChIP-on-chip data is noisy and significant effort may be needed to clean it up, the analysis we present next demonstrates that a combination of biclustering and our layout algorithm has the potential to yield biologically useful results.

The two protein-DNA datasets we study correspond to growth of *S. cerevisiae* cells in rich medium [23] and to growth under exposure to rapamycin [2], a condition that mimics nutrient starvation. We restricted our attention to transcription factors studied in both papers. We ran our implementation of the *Apriori* algorithm [1] that computes closed itemsets on both these datasets, applied our layout algorithm on biclusters with at least two genes and at least two transcription factors, and obtained the layout in Figure 4(a). Biclusters obtained from the data under growth in rich medium are shown as blue boxes and rapamycin-induced biclusters are shown as red boxes. A cell in the figure is dark grey (respectively, light grey) if the transcription factor binds to the gene's promotor in both (respectively, one) condition. The image strikingly demonstrates that under exposure to rapamycin, the transcriptional network activated in the cell is very different from the normal activation network. The rich medium data contains only four biclusters involving these transcription factors while the rapamycin data contains 38 biclusters. We conclude that very few genes are co-regulated by the same set of transcription factors in both conditions.

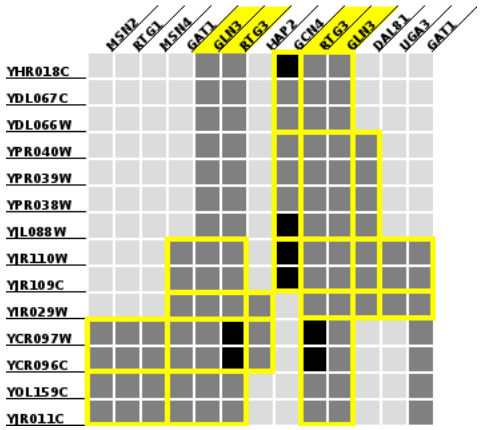
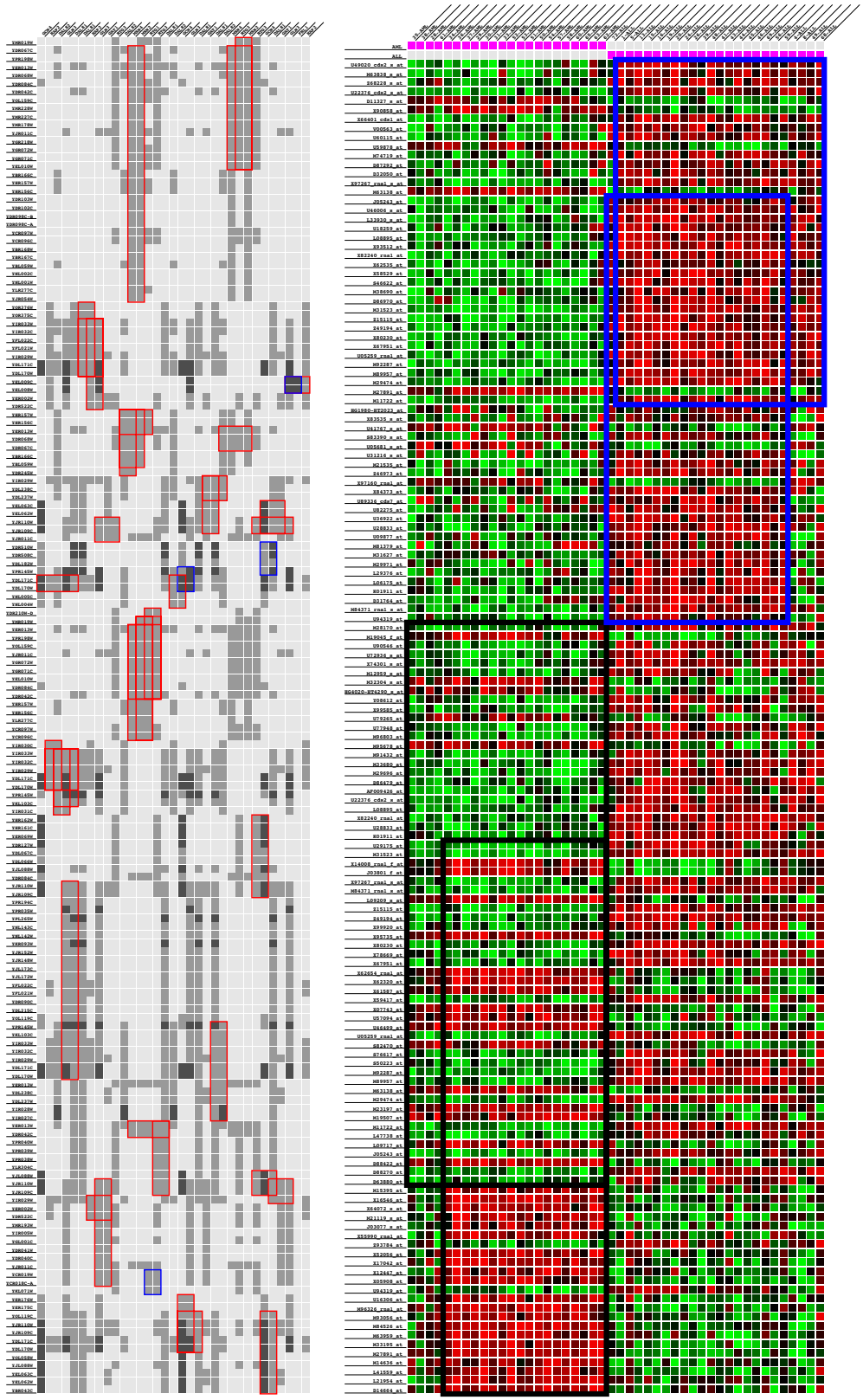


Figure 5: Genes combinatorially controlled by GLN3 and RTG3.

To illustrate the use of our web interface, we used it to search for biclusters that included the transcription factors RTG3 and GLN3. RTG3 is a transcription factor that forms a com-



(a) Combinatorial control of transcription in *S. cerevisiae*.

(b) Biclusters in gene expression data for ALL and AML.

Figure 4: Visualizations of the layouts computed by our algorithm.

plex with RTG1 to activate the retrograde (RTG) and target of rapamycin (TOR) pathways [10; 30]. GLN3 encodes a transcription factor that is phosphorylated and localised to the cytoplasm when the cell is grown in nitrogen-rich media. Rapamycin treatment can induce the dephosphorylation and subsequent activation of GLN3 [4]. Figure 5 displays the layout of all the biclusters containing these two transcription factors. We note that each bicluster also includes either the transcription factor GAT1 or the transcription factor GCN4. GAT1 is a transcriptional activator of genes involved in nitrogen catabolite repression; the activity and localization of these genes is regulated by nitrogen limitation. GCN4 is another transcription activator that is a master regulator of gene expression during amino acid starvation in *S. cerevisiae* and is activated in multiple stress responses [29]. Thus, it is not suprising that GAT1 and GCN4 co-regulate genes with GLN3 and RTG3. The functional annotations of the set of nine genes targeted by GCN4, GLN3, and RTG3 are enriched in the Gene Ontology biological process "glutamine family amino acid biosynthesis" with a p-value of 2×10^{-8} (based on the hypergeometric distribution), indicating that this pathway may be activated by the three transcription factors upon rapamycin treatment.

4.3 Classification of Leukemias

Golub et al. [15] studied global expression patterns of 45 patients diagnosed with Acute Lymphoblastic Leukemia (ALL) and 27 patients diagnosed with Acute Myeloid Leukemia (AML). We ran the xMotif algorithm [16; 28] to compute biclusters in this dataset. We ensured that computed biclusters contain samples from at most one class. We selected four representative biclusters from the results to visualize. Figure 4(b) displays the layout. Each column corresponds to a sample; the two columns at the top with purple cells indicate the type of leukemia. We map the expression values of each gene into a range from green to red, with green (respectively, red) corresponding to the smallest (respectively, largest) expression value of that gene. The biclusters outlined in black correspond to AML samples and those outlined in blue to ALL samples. This layout visually highlights similarities and differences between the biclusters found in samples for the same and for different types of leukemia. We have used such biclusters as the basis for constructing a classifier that distinguishes between different diseases and tissues [16].

5. CONCLUSIONS

The biomedical community has access to large quantities of publicly-available gene expression datasets. Biclustering has emerged as a powerful methodology for analyzing these datasets. In this paper, we have introduced a novel algorithm for laying out biclusters in a two-dimensional matrix so as to reveal the overlaps and relationships between the biclusters. The algorithm performs efficiently in practice. We have demonstrated the applicability of the algorithm to two important problems in bioinformatics using both binary and real-valued data. An easy-to-use web interface distributed with the layout software allows the user to query and navigate layouts that are too large to study manually. Biclustering is useful not just for processing gene expression data but for any dataset that measures the relationships between two different types of data, for example, genes and functions, transcription factors and promotors, microRNAs and their target mRNAs, genes and diseases, etc. Thus, our algorithm has the potential to be useful for a wide variety of bioinformatic applications.

6. REFERENCES

[1] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules in Large Databases. In *Proceedings of the Twentieth International Conference on Very Large Databases*, pages 487–499, Santiago, Chile, 1994.

[2] Z. Bar-Joseph, G. K. Gerber, T. I. Lee, N. J. Rinaldi, J. Y. Yoo, F. Robert, D. B. Gordon, E. Fraenkel, T. S. Jaakkola, R. A. Young, and D. K. Gifford. Computational discovery of gene modules and regulatory networks. *Nat Biotechnol*, 21(11):1337–42, 2003.

[3] S. Batzoglou and S. Istrail. Physical mapping with repeated probes: The hypergraph superstring problem. *Journal of Discrete Algorithms*, 1:51–76, 2000.

[4] T. Beck and M. N. Hall. The TOR signalling pathway controls nuclear localization of nutrient-regulated transcription factors. *Nature*, 402(6762):689–92, 1999.

[5] S. Bergmann, J. Ihmels, and N. Barkai. Iterative signature algorithm for the analysis of large-scale gene expression data. *Phys Rev E Stat Nonlin Soft Matter Phys*, 67(3 Pt 1):031902, 2003.

[6] S. Bergmann, J. Ihmels, and N. Barkai. Similarities and differences in genome-wide expression data of six organisms. *PLoS Biol*, 2(1):E9, 2003.

[7] K. S. Booth and G. S. Lueker. Testing for the consecutive ones property, interval graphs, and planarity using pq-tree algorithms. *J. Comput. Sys. Sci.*, 13:335–379, 1976.

[8] A. Califano, G. Stolovitzky, and Y. Tu. Analysis of gene expression microarrays for phenotype classification. *Proc Int Conf Intell Syst Mol Biol*, 8:75–85, 2000.

[9] Y. Cheng and G. Church. Biclustering of expression data. *Proc Int Conf Intell Syst Mol Biol*, 8:93–103, 2000.

[10] J. L. Crespo, T. Powers, B. Fowler, and M. N. Hall. The TOR-controlled transcription activators GLN3, RTG1, and RTG3 are regulated in response to intracellular levels of glutamine. *Proc Natl Acad Sci U S A*, 99(10):6784–9, 2002.

[11] M. Eisen, P. Spellman, P. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proc. Natl. Acad. Sci. USA*, 95:14863–14868, 1998.

[12] D. W. F. Alizadeh, R.M. Karp and G. Zweig. Physical mapping of chromosomes using unique probes. *Journal of Computational Biology*, 2:159–184, 1995.

[13] S. Fodor, R. Rava, X. Huang, A. Pease, C. Holmes, and C. Adams. Multiplexed biochemical assays with biological chips. *Nature*, 364(6437):555–6, 1993.

[14] G. Getz, E. Levine, and E. Domany. Coupled two-way clustering of dna microarray data. *Proc. Natl Acad. Sci. USA*, 97:12079–12084, 2000.

[15] T. Golub, D. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. Mesirov, H. Coller, M. Loh, J. Downing, M. Caligiuri, C. Bloomfield, and E. S. Lander. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.

[16] G. Grothaus and T. M. Murali. Biologically-interpretable disease and tissue classification based on dna microarray data. In preparation, 2006.

[17] J. A. Hartigan. Direct clustering of a data matrix. *J. Amer. Statis.*, 67:123–129, 1972.

[18] W.-L. Hsu. A simple test for the consecutive ones property. *J. Algorithms*, 43(1):1–16, 2002.

[19] T. Jiang and R. Karp. Mapping clones with a given ordering or interleaving. *Algorithmica*, 21:262–284, 1998.

[20] Y. Kluger, R. Basri, J. Chang, and M. Gerstein. Spectral biclustering of microarray data: coclustering genes and conditions. *Genome Res*, 13(4):703–16, 2003.

[21] E. Lander and M. Waterman. Genomic mapping by fingerprinting random clones: A mathematical analysis. *Genomics*, 2:231–239, 1988.

[22] L. Lazzeroni and A. Owen. Plaid models for gene expression data. *Statistica Sinica*, 12:61–86, 2002.

[23] T. I. Lee, N. J. Rinaldi, F. Robert, D. T. Odom, Z. Bar-Joseph, G. K. Gerber, N. M. Hannett, C. T. Harbison, C. M. Thompson, I. Simon, J. Zeitlinger, E. G. Jennings, H. L. Murray, D. B. Gordon, B. Ren, J. J. Wyrick, J.-B. Tagne, T. L. Volkert, E. Fraenkel, D. K. Gifford, and R. A. Young. Transcriptional Regulatory Networks in Saccharomyces cerevisiae. *Science*, 298(5594):799–804, 2002.

[24] J. Lepre, J. Rice, Y. Tu, and G. Stolovitzky. Genes@Work: an efficient algorithm for pattern discovery and multivariate feature selection in gene expression data. *Bioinformatics*, 20(7):1033–44, 2004.

[25] W.-F. Lu and W.-L. Hsu. A test for the consecutive ones property on noisy data-application to physical mapping and sequence assembly. *Journal of Computational Biology*, 10(5):709–735, 2003.

[26] S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2004. to appear.

[27] G. Mayraz and R. Shamir. Construction of physical maps from oligonucleotide fingerprints data. *RECOMB*, 1999.

[28] T. M. Murali and S. Kasif. Extracting conserved gene expression motifs from gene expression data. In *Proceedings of the Pacific Symposium on Biocomputing*, 2003. 77–88.

[29] K. Natarajan, M. R. Meyer, B. M. Jackson, D. Slade, C. Roberts, A. G. Hinnebusch, and M. J. Marton. Transcriptional profiling shows that Gcn4p is a master regulator of gene expression during amino acid starvation in yeast. *Mol Cell Biol*, 21(13):4347–68, 2001.

[30] B. A. Rothermel, J. L. Thornton, and R. A. Butow. Rtg3p, a basic helix-loop-helix/leucine zipper protein that functions in mitochondrial-induced changes in gene expression, contains independent activation domains. *J Biol Chem*, 272(32):19801–7, 1997.

[31] M. Schena, D. Shalon, R. Davis, and P. Brown. Quantitative monitoring of gene expression patterns with a complementary dna microarray. *Science*, 270(5235):467–70, 1995.

[32] E. Segal, A. Battle, and D. Koller. Decomposing gene expression into cellular processes. *Pac Symp Biocomput*, pages 89–100, 2003.

[33] E. Segal, M. Shapira, A. Regev, D. Botstein, D. Koller, and N. Friedman. Module networks: identifying regulatory modules and their condition-specific regulators from gene expression data. *Nat Genet*, 34(2):166–76, 2003.

[34] Q. Sheng, Y. Moreau, and B. De Moor. Biclustering microarray data by Gibbs sampling. *Bioinformatics*, 19 Suppl 2:II196–II205, 2003.

[35] A. Tanay, R. Sharan, M. Kupiec, and R. Shamir. Revealing modularity and organization in the yeast molecular network by integrated analysis of highly heterogeneous genomewide data. *Proc Natl Acad Sci U S A*, 101(9):2981–6, 2004.

[36] A. Tanay, R. Sharan, and R. Shamir. Discovering statistically significant biclusters in gene expression data. In *Proceedings of ISMB 2002*, pages S136–S144.

[37] A. Tanay, R. Sharan, and R. Shamir. *Handbook of Bioinformatics*, chapter Biclustering Algorithms: A Survey. 2004.

Invited Talk: Deciphering Gene Regulatory Networks by in silico approaches

Sridhar Hannenhalli
Penn Center for Bioinformatics
Department of Genetics
University of Pennsylvania
Philadelphia, PA 19104 USA
sridharh@pcbi.upenn.edu

ABSTRACT

Biological processes are controlled at various levels in the cell and while these mechanisms are poorly understood, transcriptional control is widely recognized as an important component and a better understanding of which will provide an efficient means for the therapeutic intervention in disease processes. We have been focusing on various computational problems pertaining to transcriptional regulation, namely, (1) representation and identification of transcription factor binding sites, (2) PolII promoter prediction, (3) Predicting interaction among transcription factors, (4) Transcriptional modeling, i.e. identifying arrangements of TFs that co-regulate a set of transcripts. I will present a brief overview of the computational approaches and challenges as well as a number of applications including transcriptional regulation in memory storage, heart failure, and osteoarthritis.

exMotif: Efficient Structured Motif Extraction

Yongqiang Zhang
Computer Science Department
Rensselaer Polytechnic Institute
Troy, NY 12180, USA
zhangy0@cs.rpi.edu

Mohammed J. Zaki
Computer Science Department
Rensselaer Polytechnic Institute
Troy, NY 12180, USA
zaki@cs.rpi.edu

ABSTRACT

Extracting motifs from sequences is a mainstay of bioinformatics. We look at the problem of mining structured motifs, which allow variable length gaps between simple motif components. We propose an efficient algorithm, called exMotif, that given some sequence(s), and a structured motif template, extracts all *frequent* structured motifs that have quorum q. Potential applications of our method include the extraction of single/composite regulatory binding sites in DNA sequences. exMotif is efficient in terms of both time and space and outperforms RISO, a state-of-the-art algorithm.

1. INTRODUCTION

Analyzing and interpreting sequence data is an important task in bioinformatics. One critical aspect of such interpretation is to extract important motifs (patterns) from sequences. The challenges for motif extraction problem are two-fold: one is to design an efficient algorithm to enumerate the frequent motifs; the other is to statistically validate the extracted motifs and report the significant ones.

Motifs can be classified into two main types. If no variable gaps are allowed in the motif, it is called a *simple motif*. For example, in the genome of *Saccharomyces cerevisiae*, the binding sites of transcription factor, GAL4, have as consensus [25], the simple motif, CGGN[11]CCG. Here N[11] means that there is a fixed "gap" (or don't care characters), 11 positions long. If variable gaps are allowed in a motif, it is called a *structured motif*. A structured motif can be regarded as an ordered collection of simple motifs with gap constraints between each pair of adjacent simple motifs. For example, many *retrotransposons* in the *Ty1-copia* group [17] have as consensus the structured motif: MT[115,136]MTNTAYGG[121,151]GTNGAYGAY. Here MT, MTNTAYGG and GTNGAYGAY are three simple motifs; [115,136] and [121,151] are variable gap constraints ([minimum gap, maximum gap]) allowed between the adjacent simple motifs. More formally, a structured motif, $\mathcal{M}$, is specified in the form:

$$M_1[l_1, u_1]M_2[l_2, u_2]M_3 \ldots M_{k-1}[l_{k-1}, u_{k-1}]M_k$$

where M_i, $1 \leq i \leq k$, is a simple motif *component*; and l_i and u_i, $1 \leq i < k$, are the minimum and maximum number of gaps allowed between M_i and M_{i+1}, respectively. The number of simple motif components, k, is also called the *length* of $\mathcal{M}$. Let W_i, $1 \leq i < k$, denote the span of the gap range, $[l_i, u_i]$, which is calculated as: $W_i = u_i - l_i + 1$.

In the structured motif extraction problem, the component motifs M_i are *unknown* before the extraction. However, we do provide some *known* parameters to restrict the structured motifs to be extracted, including: (i) k – the *length* of $\mathcal{M}$; (ii) $|M_i|$ – the length of each component $M_i \in \mathcal{M}$, for $1 \leq i \leq k$; and (iii) $[l_i, u_i]$ – the gap range between M_i and M_{i+1}, for $1 \leq i < k$. All these parameters define a *structured motif template*, $\mathcal{T}$, for the structured motifs to be extracted from a set of sequences $\mathcal{S}$. A structured motif $\mathcal{M}$ matching the template $\mathcal{T}$ in $\mathcal{S}$ is called an *instance* of $\mathcal{T}$.

Let $\delta_S(\mathcal{M})$ denote the number of occurrences of an instance motif $\mathcal{M}$ in a sequence $S \in \mathcal{S}$. Let $d_S(\mathcal{M}) = 1$ if $\delta_S(\mathcal{M}) > 0$ and $d_S(\mathcal{M}) = 0$ if $\delta_S(\mathcal{M}) = 0$. The *support* of motif $\mathcal{M}$ in the is defined as $\pi(\mathcal{M}) = \sum_{S \in \mathcal{S}} d_S(\mathcal{M})$, i.e., the number of sequences in $\mathcal{S}$ that contain at least one occurrence of $\mathcal{M}$. The *weighted support* of $\mathcal{M}$ is defined as $\pi_w(\mathcal{M}) = \sum_{S \in \mathcal{S}} \delta_S(\mathcal{M})$, i.e., total number of occurrences of $\mathcal{M}$ over all sequences in $\mathcal{S}$. We use $\mathcal{O}(\mathcal{M})$ to denote the set of all occurrences of a structured motif $\mathcal{M}$. Given a user-specified quorum threshold $q \geq 2$, a motif that occurs at least q times will be called *frequent*.

There are two main tasks in the structured motif extraction problem: a) *Common Motifs* – find all motifs $\mathcal{M}$ in a set of sequences $\mathcal{S}$, such that the support of $\mathcal{M}$ is at least q, b) *Repeated Motifs* – find all motifs in a single sequence S, such that the weighted support of $\mathcal{M}$ is at least q. Furthermore, the structured motif extraction problem allows several variations:

- *Substitutions:* $\mathcal{O}$ may consist of similar motifs, as measured by *Hamming Distance* ([13]), instead of exact matches, to the simple motifs in $\mathcal{M}$. We can either allow for at most ϵ_i errors for each simple motif M_i, $1 \leq i \leq k$, or at most ϵ errors for the whole structured motif $\mathcal{M}$.
- *Overlapping Motifs:* The variable gap constraints (l_i and u_i) can take on *negative* values, allowing extraction of overlapping simple motifs.
- *Motif Length Ranges:* Each simple motif M_i in a template $\mathcal{M}$ can be of a range of lengths, i.e., $|M_i| \in [l_a, l_b]$, where l_a and l_b are the lower and upper bounds on the desired length.

Table 1 shows four example DNA sequences $S_1, S_2, S_3, S_4 \in \mathcal{S}$; a structured motif template $\mathcal{T}$, where M_1 = NNN, M_2 = NN and M_3 = NNNN ('N' stands for any of the four DNA

Table 1: Structured Motif Extraction

Sequence S_1 ($\in \mathcal{S}$):	**CCGTA**CC**GAAC**CTCAAA
Sequence S_2 ($\in \mathcal{S}$):	**CCGT**TAT**A**G**GAAC**CATT
Sequence S_3 ($\in \mathcal{S}$):	TAT GGAACCATCTT
Sequence S_4 ($\in \mathcal{S}$):	TAACGGATCCCTTT
Structured Motif Template ($\mathcal{T}$):	NNN[0,3]NN[1,3]NNNN
Quorum (q):	2

bases: A,C,G,T), and [0,3] and [1,3] are the intervening gap ranges between the components; and a quorum threshold $q = 2$. The length of the template $\mathcal{T}$ is $k = 3$. The span of gap ranges are: $W_1 = u_1 - l_1 + 1 = 2$ and $W_2 = u_2 - l_2 + 1 = 2$. If no substitutions are allowed, there are five frequent structured motifs in $\mathcal{S}$ matching the template$\mathcal{T}$, namely $\mathcal{M}_1$ = CCG[0,3]TA[1,3]GAAC (shown in bold) and $\mathcal{M}_2$ = CCG[0,3]TA[1,3]AACC which occur in S_1 and S_2; $\mathcal{M}_3$ = TAT[0,3]GG[1,3]ACCA (shown underlined), $\mathcal{M}_4$ = TAT[0,3]GA[1,3]CCAT and $\mathcal{M}_5$ = TAT[0,3] GG[1,3]CCAT which occur in S_2 and S_3. If substitutions are allows, say, $e_1 = 1 = e_3$, then the occurrence of $\mathcal{M}_6$ = TAA[0,3]GG[1,3] CCCT (shown underlined) in S_4 will be considered to match motif $\mathcal{M}_5$.

In this paper, we propose exMotif, an efficient algorithm for both the structured motif extraction problems. It uses an inverted index of symbol positions, and it enumerates all structured motifs by *positional joins* over this index. The variable gap constraints are also considered at the same time as the joins, resulting in considerable efficiency. In order to save time and space, we only keep the start positions of each intermediate pattern during the positional join.

2. RELATED WORK

Many simple motif extraction algorithms have been proposed primarily for extracting the transcription factor binding sites, where each motif consists of a unique binding site [19; 20; 15; 14; 3; 18; 11] or two binding sites separated by a fixed number of gaps [22; 10; 9]. A pattern with a single component is also called a *monad pattern*. Structured motif extraction problems, in which variable number of gaps are allowed, have attracted much attention recently, where the structured motifs can be extracted either from multiple sequences [12; 6; 7; 16; 8; 5; 2; 1] or from a single sequence [24; 4]. In many cases, more than one transcription factor may cooperatively regulate a gene. Such patterns are called *composite regulatory patterns*. To detect the composite regulatory patterns, one may apply single binding site identification algorithms to detect each component separately. However, this solution may fail when some components are not very strong (significant). Thus it is necessary to detect the whole composite regulatory patterns (even with weak components) directly, whose gaps and other possibly strong components can increase its significance.

Several algorithms have been used to address the composite pattern discovery with two components, which are called *dyad patterns*. Helden et al. [22] propose a method for dyad analysis, which exhaustively counts the number of occurrences of each possible pair of patterns in the sequences and then assesses their statistical significance. This method can only deal with fixed number of gaps between the two components. MITRA [10] first casts the composite pattern discovery problem as a larger monad discovery problem and then applies an exhaustive monad discovery algorithm. It can handle several mismatches but can only handle sequences less than 60 kilo-bases long. Co-Bind [21] models composite transcription factors with Position Weight Matrices (PWMs) and finds PWMs that maximize the joint likelihood of occurrences of the two binding site components. Co-Bind uses Gibbs sampling to select binding sites and then refines the PWMs for a fixed number of times. Co-Bind may miss some binding sites since not all patterns in the sequences are considered. Moreover, using a fixed number of iterations for improvement may not converge to the global optimal dyad PWM.

SMILE [12] describes four variants of increasing generality for common structured motif extraction, and proposes two solutions for them. The two approaches for the first problem, in which the structured motif template consists of two components with a gap range between them, both start by building a generalized suffix tree for the input sequences and extracting the first component. Then in the first approach, the second component is extracted by simply jumping in the sequences from the end of the first one to the second within the gap range. In the second approach, the suffix tree is temporarily modified so as to extract the second component from the modified suffix tree directly. The drawback of SMILE is that its time and space complexity are exponential in the number of gaps between the two components. In order to reduce the time during the extraction of the structured motifs, [8] presents a parallel algorithm, PSmile, based on SMILE, where the search space is well-partitioned among the available processors.

RISO [6; 7; 16] improves SMILE in two aspects. First, instead of building the whole suffix tree for the input sequences, RISO builds a suffix tree only up to a certain level l, called a*factor tree*, which leads to a large space saving. Second, a new data structure called *box-link* is proposed to store the information about how to jump within the DNA sequences from one simple component (box) to the subsequent one in the structured motif. This accelerates the extraction process and avoids exponential time and space consumption (in the gaps) as in SMILE. In RISO, after the generalized factor tree is built, the box-links are constructed by exhaustively enumerating all the possible structured motifs in the sequences and are added to the leaves of the factor tree. Then the extraction process begins during which the factor tree may be temporarily and partially modified so as to extract the subsequent simple motifs. Since during the box-link construction, the structured motif occurrences are exhaustively enumerated and the frequency threshold is never used to prune the candidate structured motifs, RISO needs a lot of computation during this step.

For repeated structured motif identification problem, the

Table 2: Pos-lists (Sequence identifiers (i) and cardinality of $\mathcal{P}(X, S_i)$ are marked in bold)

X	*pos-lists*
A	{**1**,**6**,5,9,10,15,16,17, **2**,**5**,6,8,11,12,15, **3**,**4**,2,6,7,10, **4**,**3**,2,3,7}
C	{**1**,**7**,1,2,6,7,11,12,14, **2**,**4**,1,2,13,14, **3**,**3**,8,9,12, **4**,**4**,4,9,10,11}
G	{**1**,**2**,3,8, **2**,**3**,3,9,10, **3**,**2**,4,5, **4**, **2**,5,6}
T	{**1**,**2**,4,13, **2**,**5**,4,5,7,16,17 **3**,**5**,1,3,11,13,14, **4**,**5**,1,8,12,13,14}

frequency closure property that "all the subsequences of a frequent sequence must be frequent", doesn't hold any more since the frequency of a pattern can exceed the frequency of its sub-patterns. [24] introduces an closure-like property which can help prune the patterns without missing the frequent patterns. The two algorithms proposed in [24] can extract within one sequence all frequent patterns of length no greater than a length threshold, which can be either manually specified or automatically determined. However, the gap ranges between adjacent symbols are required to be same, and approximate matches are not allowed.

3. THE EXMOTIF ALGORITHM

We first introduce our basic approach for common structured motif extraction problem. We then successively optimize it for various practical scenarios.

3.1 The Basic Approach

Let's assume that we are extracting all structured motif instances from n sequence $\mathcal{S} = \{S_i, 1 \leq i \leq n\}$, each of which satisfies the template $\mathcal{T}$ and occurs at least in q sequences of $\mathcal{S}$. We assume for the moment that no substitutions are allowed in any of the simple motifs. We also assume that all $S_i \in \mathcal{S}, 1 \leq i \leq n$ and the extracted motifs are over the DNA alphabet, Σ_{DNA}. EXMOTIF first converts each $S_i \in \mathcal{S}, 1 \leq i \leq n$ into an equivalent inverted format [23], where we associate with each symbol in the sequence S_i its *pos-list*, a *sorted* list of the positions where the symbol occurs in S_i. Then for each symbol we combine its pos-list in each S_i to obtain its pos-list in $\mathcal{S}$. More formally, for a symbol $X \in \Sigma_{\text{DNA}}$, its pos-list in S_i is given as $\mathcal{P}(X, S_i) = \{j \mid S_i[j] = X, j \in [1, |S_i|]\}$, where $S_i[j]$ is the symbol at position j in S_i, and $|S_i|$ denotes the length of S_i. Its pos-list across all sequences $\mathcal{S}$ is obtained by grouping the pos-lists of each sequence, and is given as $\mathcal{P}(X, \mathcal{S}) = \{\langle i, |\mathcal{P}(X, S_i)|, \mathcal{P}(X, S_i)\rangle \mid S_i \in \mathcal{S}\}$, where i is the *sequence identifier* of S_i, and $|\mathcal{P}(X, S_i)|$ denotes the cardinality of the pos-list $\mathcal{P}(X, S_i)$ in sequence S_i. For our example sequences in Table 1, the pos-list for each DNA base is given in Table 2. For example, A occurs in sequence S_1 at the positions $\{5, 9, 10, 15, 16, 17\}$, thus the entries in A's pos-list are $\{\mathbf{1}, \mathbf{6}, 5, 9, 10, 15, 16, 17\}$.

3.1.1 Positional Joins

We first extend the notion of pos-lists to cover structured motifs. The pos-list of $\mathcal{M}$ in $S_i \in \mathcal{S}$ is given as the set of start positions of all the matches of $\mathcal{M}$ in S_i. Let $X, Y \in \Sigma_{\text{DNA}}$ be any two symbols, and let $\mathcal{M} = X[l, u]Y$ be a structured motif. Given the pos-lists of X and Y in S_i for $1 \leq i \leq n$, namely, $\mathcal{P}(X, S_i)$ and $\mathcal{P}(Y, S_i)$, the pos-list of $\mathcal{M}$ in S_i can be obtained by a positional join as follows: for a position $x \in \mathcal{P}(X, S_i)$, if there exists a position $y \in \mathcal{P}(Y, S_i)$, such that $l \leq y - x - 1 \leq u$, it means that Y follows X within the variable gap range $[l, u]$ in the sequence S_i, and thus we can add x to the pos-list of motif $X[l, u]Y$. Let d be the number of gaps between $x \in \mathcal{P}(X, S_i)$ and $y \in \mathcal{P}(Y, S_i)$, given as $d = y - x - 1$. Then, in general, there are three cases to consider in the positional join algorithm:

- $d < l$: Advance y to the next element in $\mathcal{P}(Y, S_i)$.
- $d > u$: Advance x to the next element in $\mathcal{P}(X, S_i)$.
- $l \leq d \leq u$: Save this occurrence in $\mathcal{P}(X[l, u]Y, S_i)$, and then advance x to the next element in $\mathcal{P}(X, S_i)$.

The pos-list for $X[l, u]Y$ can be computed in time linear in the lengths of $\mathcal{P}(X, S_i)$ and $\mathcal{P}(Y, S_i)$. In essence, each time we advance $x \in \mathcal{P}(X, S_i)$, we check if there exists a $y \in \mathcal{P}(Y, S_i)$ that satisfies the given gap constraint. Instead of searching for the matching y from the beginning of the pos-list each time, we search from the last position used to compare with x. This results in fast positional joins. For example, during the positional join for the motif A[0,1]T in S_4, with $l = 0$ and $u = 1$, we scan the pos-lists of A and T for S_4 in Table 2, i.e. $\mathcal{P}(X, S_4) = \{2, 3, 7\}$ and $\mathcal{P}(Y, S_4) = \{1, 8, 12, 13, 14\}$. Initially, $x = 2$ and $y = 1$. This gives $d = 1 - 2 - 1 = -2 < l$, thus we advance y to 8. Next, $d = 8 - 2 - 1 = 5 > u$, thus we advance x to 3. Then, $d = 8-3-1 = 4 > u$, thus we still advance x to 7. Next, $d = 8-7-1 = 0 \in [l, u]$, so we store $x = 7$ in $\mathcal{P}(A[0, 1]T, S_4)$. We would advance x but since we have already reached the end of $\mathcal{P}(A, S_4)$, the positional join stops. Thus the final pos-list of A[0,1]T in S_4 is: $\mathcal{P}(A[0, 1]T, S_4) = \{7\}$. After we obtain the pos-list of $\mathcal{M}$ in each S_i for $1 \leq i \leq n$, we can combine them together to obtain the pos-list of $\mathcal{M}$ in $\mathcal{S}$. For example, the full pos-list of A[0,1]T for $\mathcal{S}$ is: {**2**,**2**,6,15, **3**,**2**,2,10, **4**,**1**,7}. Thus the support of A[0,1]T is 3. Note here for each non-empty pos-list, we insert its sequence identifier and length before it.

Given a longer motif $\mathcal{M}$, the positional joins start with the last two symbols, and proceed by successively joining the pos-list of the current symbol with the intermediate pos-list of the suffix. That is, the intermediate pos-list for a *(l+1)*-length pattern (with $l \geq 1$) is obtained by doing a positional join of the pos-list of the pattern's first symbol, called the *head symbol*, with the pos-list of its *l*-length suffix, called the *tail*. As the computation progresses the previous tail pos-lists are discarded. Combined with the fact that only start positions are kept in a pos-list, this saves both time and space.

In order to enumerate all frequent motifs instances $\mathcal{M}$ in $\mathcal{S}$, EXMOTIF computes the pos-list for each $\mathcal{M}$ and report $\mathcal{M}$ only if its support is no less than the quorum (q). A straightforward approach is to directly perform positional joins on the symbols from the end to the start for each $\mathcal{M}$. This approach leads to much redundant computation

since simple motif components may be shared among several structured motifs. exMotif, in contrast, performs two steps: it first computes the pos-lists for all simple motifs in $\mathcal{S}$ by doing positional joins on pos-lists of its symbols (as described in Section *3.1.2*), and it then computes the pos-list for each structured motif by doing positional joins on pos-lists of its simple motif components (as described in Section *3.1.3*). exMotif handles both simple and structured motifs uniformly, by adding the gap range $[0, 0]$ between adjacent symbols within each simple motif M_i. For our example in Table 1, the structured motif template $\mathcal{T}$ becomes: N[0,0]N[0,0]N[0,1]N[0,0]N[2,3]N[0,0]N[0,0]N[0,0]N. Also since we only report frequent motifs, we can prune the candidate patterns during the positional joins based on the closure property of support (note however that this cannot be done for weighted support).

3.1.2 Extraction of the Simple Motifs

Given a template motif $\mathcal{T}$, we know the lengths of the simple motif components desired. A naive approach is to directly do positional joins on the symbols from the end to the start of each simple motif. However, since some simple motifs are of the same length and the longer simple motifs can be obtained by doing positional joins on the shorter simple motifs/symbols, we can avoid some redundant computation. Note also that the gap range inside the simple motif is always [0,0].

Let $\mathcal{L} = \{L_i, 1 \leq i \leq m\}$, where L_i is the length of each simple motif in $\mathcal{T}$ and assume $\mathcal{L}$ is sorted in the ascending order. For each $L_i, 1 \leq i \leq m$, we need to enumerate $|\Sigma_{\text{DNA}}|^{L_i}$ possible simple motifs. Let $max_{\mathcal{L}}$ be the maximum length in $\mathcal{L}$. We can compute the pos-lists of simple motifs sequentially from length 1 to $max_{\mathcal{L}}$. But this may waste time in enumerating some simple motifs of lengths that are not in $\mathcal{L}$. Instead, exMotif first computes the pos-lists for the simple motifs of lengths that are powers of 2. Formally, let J be an integer such that $2^J \leq max_{\mathcal{L}} < 2^{J+1}$. We extract the patterns of length 2^j by doing positional joins on the *pos-list*s of patterns of length 2^{j-1} for all $1 \leq j \leq J$. For example, when $max_{\mathcal{L}} = 11$, exMotif first computes the pos-lists for simple motifs of length $2^0 = 1$, $2^1 = 2$, $2^2 = 4$ and $2^3 = 8$. exMotif then computes the pos-lists for the simple motifs of $L_i \in \mathcal{L}$, by doing positional joins on simple motifs whose *pos-list*(s) have already been computed and their lengths sum to L_i. For example, when $L_i = 11$, exMotif has to join motifs of lengths 8, 2, and 1. It first obtains all motifs of length 8+2=10, and then joins the motifs of lengths 10 and 1, to get the pos-lists of all simple motifs of length 10+1=11. At the end of the first phase, exMotif has computed the pos-lists for all simple motif components that can satisfy the template.

3.1.3 Extraction of the Structured Motifs

We extract the structured motifs by doing positional joins on the pos-lists of the simple motifs from the end to the start in the structured motif $\mathcal{M}$. Formally, let $H[l, u]T$ be an intermediate structured motif, with simple motif H as the head, and a suffix structured motif T as tail. Then $\mathcal{P}(H[l, u]T)$ can be obtained by doing positional joins on $\mathcal{P}(H)$ and $\mathcal{P}(T)$. Since $\mathcal{P}(H)$ keeps only the start positions, we need to compute the corresponding end positions for those occurrences of H, to check the gap constraints. Since only exact matches or substitutions are allowed for simple motifs, the end position is simply $s + |H| - 1$ for a start position s.

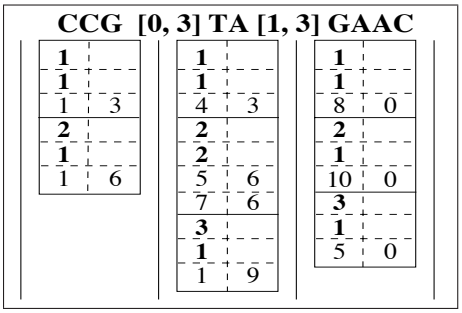


Figure 1: Indexed Full-position Recovery

3.1.4 Full-position Recovery

In our positional join approach, to save time and space we retain only the motif start positions, however, in some applications, we may need to know the full position of each occurrence, i.e., the set of matching positions for each symbol in the motif. In exMotif we "index" some information during the positional joins in order to facilitate full position recovery. Consider the example shown in Fig. 1 to recover the full positions for $\mathcal{M}$ = CCG[0,3]TA[1,3]GAAC. Under each symbol we show two columns. The left column corresponds to the intermediate pos-lists as we proceed from right to left, whereas the right column stores the indices into the previous pos-list. For example, the middle column gives the pos-list $\mathcal{P}(TA[1,3]GAAC) = \{\mathbf{1}, \mathbf{1}, 4,\ \mathbf{2}, \mathbf{2}, 5, 7,\ \mathbf{3}, \mathbf{1}, 1\}$. For each position $x \in \mathcal{P}(TA[1,3]GAAC)$ (excluding the sequence identifiers and the cardinality), the right column records an index in $\mathcal{P}(GAAC)$ which corresponds to the first position in $\mathcal{P}(GAAC)$ that satisfies the gap range with respect to x. For example, for position $x = 5$ (at index 6), the first position in $\mathcal{P}(GAAC)$ that satisfies the gap range [1,3] is 10 (since in this case there are 3 gaps between the end of TA at position 6 and start of GAAC at position 10), and it occurs at index 6. Likewise, for each position in the current pos-list we store which positions in the previous pos-list were extended. With this indexed information, full-position recovery becomes straightforward. We begin with the start positions of the occurrences. We then keep following the indices from one pos-list to the next, until we reach the last pos-list. Since the index only marks the first position that satisfies the gap range, we still need to check if the following positions satisfy the gap range. At each stage in the full position recovery, we maintain a list of intermediate position prefixes $\mathcal{F}$ that match up to the j-th position in $\mathcal{M}$. For example, to recover the full position for $\mathcal{M}$ = CCG[0,3]TA[1,3]GAAC, considering start position 1 (with $\mathcal{F} = \{(1)\}$) in sequence 2, we follow index 6 to get position 5 in the middle pos-list, to get $\mathcal{F} = \{(1, 5)\}$. Since the next position after 5 is 7 which is also within the gap range [0,3], so we update $\mathcal{F} = \{(1, 5), (1, 7)\}$. For position 5, we follow index 6 to get position 10 in the rightmost pos-list, to get $\mathcal{F} = \{(1, 5, 10)\}$; for position 7, we follow index 6 to get position 10 in the right pos-list, to get $\mathcal{F} = \{(1, 7, 10)\}$. Likewise, we can recover the full-position in sequence 1, which is $\mathcal{F} = \{(1, 4, 8)\}$. During the full-position recovery, we can also count the number of full-positions, i.e., occurrences, of each structured motif. For example, there are 3 occurrences of CCG[0,3]TA[1,3]GAAC.

3.1.5 Overlapped Motifs & Length Ranges for Simple Motifs

Our positional join approach can automatically handle overlapping simple motifs, by simply adjusting the minimum gap value (i.e., allowing negative values). For example, a motif template like $M_1[-6, 10]M_2$ allows M_2 to overlap M_1 by 6 positions.

exMotif also allows variation in the lengths of the simple motifs to be found. For example, a motif template may be specified as $M_1[5, 10]M_2$, $|M_1| \in [2, 4]$, and $|M_2| \in [6, 7]$, which means that we have to consider NN, NNN, and NNNN as the possible templates for M_1 and similarly for M_2. A straightforward way for handling length ranges is to enumerate exhaustively all the possible sub-templates of $\mathcal{T}$ with simple motifs of fixed lengths and then to extract each sub-template separately. Instead, exMotif does an optimized extraction. exMotif reuses the partial pos-lists created when using a depth first search to enumerate and extract the sub-templates.

3.2 Handling Substitutions

As mutations are a common phenomena in biological sequences, we allow substitutions in the extracted motifs. That is two motif instances may be considered to be the same if they are within the allowed substitution thresholds. exMotif allows users to specify the number of substitutions allowed for the whole motif (ϵ), and also a per simple motif threshold (ϵ_i, $i \in [1, k]$). There are two types of substitutions we consider.

3.2.1 Position-Specific Substitutions

Table 3: **IUPAC Alphabet (Σ_{IUPAC})**

Symbol	A	C	G	T
Bases	A	C	G	T
Symbol	U	R	Y	K
Bases	U	A,G	C,T	G,T
Symbol	M	S	W	B
Bases	A,C	G,C	A,T	C,G,T
Symbol	D	H	V	N
Bases	A,G,T	A,C,T	A,C,G	A,C,G,T

Here we allow a position (a DNA symbol) in the instance motif $\mathcal{M}$ to be substituted with 1 or 2 other DNA symbols. All such neighbors will contribute to the frequency of $\mathcal{M}$. For example, for $\mathcal{M} = ACG[4, 6]TT$, if we allow $e_1 = 1$ substitutions in motif $M_1 = ACG$, at position 2, then $AAG[4, 6]TT$, $ACG[4, 6]TT$ or $AGG[4, 6]TT$ may contribute to the frequency of $\mathcal{M}$. Instead of enumerating all of these separately, exMotif can directly mine relevant motifs using IUPAC symbols (see Table 3). exMotif simply constructs the pos-lists for the relevant IUPAC symbols by scanning sequences in $\mathcal{S}$ once. Then it mines the motif instances as in the basic approach, since all allowed substitutions have already been incorporated into the relevant IUPAC symbols. For example, if only $e_1 = 1$ substitution is allowed in the motif, then exMotif adds R,Y,K,M,S, and W as basic symbols. Thus instead of reporting $\mathcal{M} = ACG[4, 6]TT$ as the instance, exMotif may report $ASG[4, 6]TT$ as an instance, where S stands for either C or G (see Table 3). exMotif also allows the user to specify the maximum number of IUPAC symbols that can appear in a motif.

3.2.2 Arbitrary Substitutions

Here we allow a DNA symbol in $\mathcal{M}$ to be substituted with other symbols across all positions (i.e., in a position independent manner), up to the allowed maximum errors per motif (or per component). To count the support for a motif, exMotif has to consider all of its *neighbors* as well, which are defined as all the motifs (including itself) within *Hamming distance*, ϵ (or per motif e_i). Then the support of an instance motif is calculated as the total number of sequences in which its neighbors (including itself) are present. As always, the motif is frequent if its support meets the quorum q, that is, its neighbors are present in at least q distinct sequences.

The main challenge is that when arbitrary, position independent substitutions are allowed, we cannot do support checking during each positional join, since the support of the current motif may be below quorum, but combined with its neighbors it may meet quorum. Thus exMotif does support checking at two points. First, it checks for quorum after the pos-lists of all the simple motifs in $\mathcal{T}$ have been computed, provided the per motif error thresholds e_i have been specified. In this case each simple motif must be frequent to be extended to a structured motif. Second, it checks for quorum after the pos-lists of all the structured motifs that satisfy $\mathcal{T}$ are computed. Only the frequent instances are reported.

3.2.2.1 Determining Neighbors.

In order to quickly find all the existing neighbors of a motif within the allowed error thresholds, exMotif first computes all the exact structured motifs, and stores them into a hash table to facilitate fast lookup. Then for each extracted structured motif $\mathcal{M}$, exMotif enumerates all its possible neighbors and checks whether they exist in the hash table. One problem is that the number of possible neighbors of $\mathcal{M}$ can be quite large. When we allow ϵ_i substitutions for simple component M_i in $\mathcal{M}$, for $1 \leq i \leq k$, the number of $\mathcal{M}$'s neighbors is given as $\prod_{i=1}^{k}[\sum_{j=0}^{e_i}\binom{|M_i|}{j} \cdot 3^j]$. For example, for $\mathcal{M}$ = AACGTT[1,5]AGTTCC, when we allow one substitution for each simple motif, the number of its neighbors is 361; when we allow two substitutions per component, the number of its neighbors is 23,716. Instead of enumerating the potentially large number of neighbors (many of which may not even occur in the sequence set $\mathcal{S}$) for each structured motif $\mathcal{M}$ individually, exMotif utilizes the observation that many motifs have shared neighbors, and thus previously computed support information can be reused. exMotif enumerates neighbors in two steps. In the first step, for each $\mathcal{M}$, it enumerates *aggregate* neighbor motifs, replacing the allowed number of errors e_i with as many 'N' symbols (which stands for A,C,G, or T). The number of possible aggregate neighbors is given as $\prod_{i=1}^{k}\binom{|M_i|}{\epsilon_i}$. In the second step, it computes the support for each aggregate neighbor by expanding each 'N' with each DNA symbol, looking up the hash table for the support of the corresponding motif, and adding the supports for all matching motifs. Since the motifs matching an aggregate are also neighbors of each other, the support of the aggregate can be re-used to compute the support of other matching motifs as well. Once the supports for all aggregate neighbors have been computed, the final support of the structured motif $\mathcal{M}$ can be obtained. Thus for each $\mathcal{M}$, the number of "neighbors"

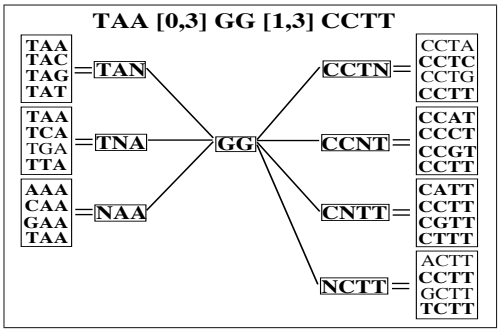


Figure 2: Aggregate Neighbors

to consider can be as low as $\prod_{i=1}^{k} \binom{|M_i|}{\epsilon_i}!$

For example, consider the example shown in Figure 2. Consider the structured motif $\mathcal{M}$ = TAA[0,3]GG[1,3]CCTT (taken from our example in Table 1); assume that $\epsilon_1 = 1$, $\epsilon_2 = 0$ and $\epsilon_3 = 1$. There are three possible aggregates for TAA, namely TAN, TNA, and NAA, and four aggregates for CCTT, namely CCTN, CCNT, CNTT, and NCTT, giving a total of 12 aggregate neighbors for $\mathcal{M}$, as illustrated in the figure. exMotif processes each aggregate neighbor in turn. Using a hash-table (or direct lookup table if there are only a few neighbors), it checks if the aggregate neighbor has been processed previously. If yes, it moves on to the next aggregate. If not, it gathers the support information from all of its matching structured motifs, to compute its total support. Next, it also updates the neighbor support value for each of the matching motifs, so that once an aggregate is processed, we no longer require its information. All we need to know is whether it has been processed or not. For example, once the support of the first aggregate TAN[0,3]GG[1,3]CCTN for the example motif $\mathcal{M}$ above is computed, exMotif also updates the neighbor supports for all other matching structured motifs, such as $\mathcal{M}'$ = TAC[0,3]GG[1,3]CCTG. Later when processing $\mathcal{M}'$, exMotif can skip the above aggregate and focus on the not yet processed aggregates, e.g., NAC[0,3]GG[1,3]NCTG, and so on.

3.2.2.2 *Counting Support.*

There are two methods to record the support for each motif. In the first method, we associate each motif with a bit vector, $\mathcal{V}$. Each bit, $\mathcal{V}_i$ for $1 \leq i \leq n$ (where $n = |\mathcal{S}|$) indicates whether the motif is present in the sequence $S_i \in \mathcal{S}$. The support of the motif is the number of set bits in $\mathcal{V}$. Thus to obtain the support for a motif, we can simply union the bit vectors of all its (aggregate) neighbors. Using one bit to represent a sequence saves space, and also saves time via the union operation. However, since we need n fixed bits for each motif to store its bit vector, this is not efficient if there are many sequences, and if a motif occurs only in a small number of sequences, which leads to a sparse bit vector. Thus in the second method, exMotif associates each motif with an identifier array, $\mathcal{A}$, to only store the sequence identifiers in which the motif occurs. exMotif can then obtain the support for a motif by scanning the identifier arrays of its neighbors in linear time. For example consider again our motif (from Table 1), TAT[0,1]GG[2,3]CCAT, which occurs in S_2 and S_3, Its bit vector is thus $\mathcal{V} = \{0110\}$ and its identifier array $\mathcal{A} = \{2, 3\}$.

3.3 Solving Repeated Structured Motif Identification Problem

In repeated structured motif identification problem, the frequency closure property (that all the subsequences of a frequent sequence must be frequent), does not hold any more. For example, the sequence GCTTT, has three occurrences of pattern G[1,3]T, but it sub-pattern, G, has only one occurrence. Thus we can't apply the closure property for pruning candidates. Nevertheless, a bound on the frequency of a sub-pattern can be established, which can be used for pruning.

Theorem 1 1. *Let $\mathcal{M} = M_1...M_k$ be a structured motif and $\mathcal{M}' = M_i...M_k$ be a suffix of $\mathcal{M}$, for $1 \leq i \leq k$. If the weighted support of $\mathcal{M}$ is $\pi_w(\mathcal{M})$, then $\pi_w(\mathcal{M}') \geq \frac{\pi_w(\mathcal{M})}{\prod_{m=1}^{i-1} W_m}$, where $W_m = u_m - l_m + 1$ is the span of the gap range for $m \in [1, k-1]$.*

Proof. Let $\mathcal{O}(\mathcal{M})$ be the occurrence set of $\mathcal{M}$ and $\mathcal{O}(\mathcal{M}')$ be the occurrence set of $\mathcal{M}'$. For each occurrence of $\mathcal{M}'$ in $\mathcal{O}(\mathcal{M}')$, we can extend it to get occurrences of $\mathcal{M}$ in $\mathcal{O}(\mathcal{M})$ by adding $M_1 \dots M_{i-1}$ before $\mathcal{M}'$. This leads to at most $\prod_{m=1}^{i-1} W_m$ occurrences of $\mathcal{M}$ for any occurrence of $\mathcal{M}'$. Thus $|\mathcal{O}(\mathcal{M}')| \cdot \prod_{m=1}^{i-1} W_m \geq |\mathcal{O}(\mathcal{M})|$, which immediately gives $\pi_w(\mathcal{M}') \geq \frac{\pi_w(\mathcal{M})}{\prod_{m=1}^{i-1} W_m}$. □

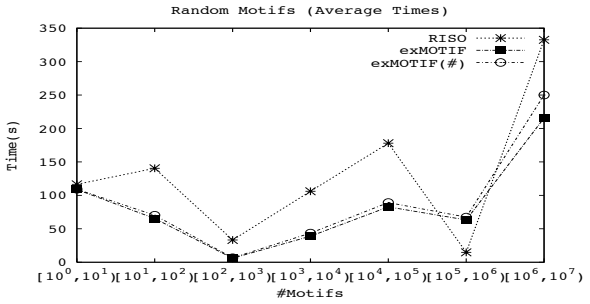
With Theorem 1, exMotif can calculate a support bound for any suffix $\mathcal{M}'$ of $\mathcal{M}$, given the quorum requirement q. For example, assume that the motif template is NN[3,5]NNN [0,4]NNN and $q = 100$, with $W_1 = 5 - 3 + 1 = 3$ and $W_2 = 4 - 0 + 1 = 5$. When processing the suffix component $\mathcal{M}'$ = NNN, we require that $\pi_w(\mathcal{M}') \geq \frac{100}{3 \times 5} = 6$; when processing $\mathcal{M}'$ = NNN[0,4]NNN, we require that $\pi_w(\mathcal{M}') \geq \frac{100}{3} = 33$. Thus even the weaker bounds can lead to some pruning.

4. EXPERIMENTAL RESULTS

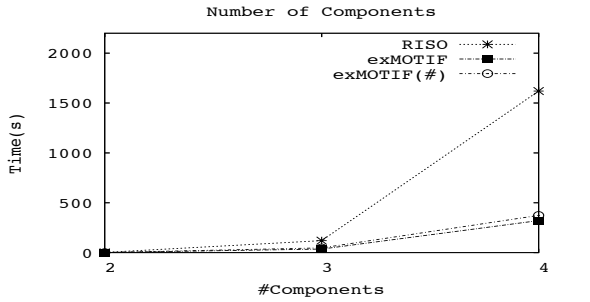
exMotif has been implemented in C++, and compiled with g++ v4.0.0 at optimization level 3 (-O3). We performed experiments on a Macintosh PowerPC G5 with dual 2.7GHz processors and 4GB memory running Mac OS X v10.4.5. We compare our results with the latest version of RISO [6; 7; 16] (called RISOTTO [16]; obtained from `http://algos.inesc-id.pt/~asmc/software/riso.html`), the best previous algorithm for structured motif extraction problem.

4.1 exMotif and RISO: Comparison

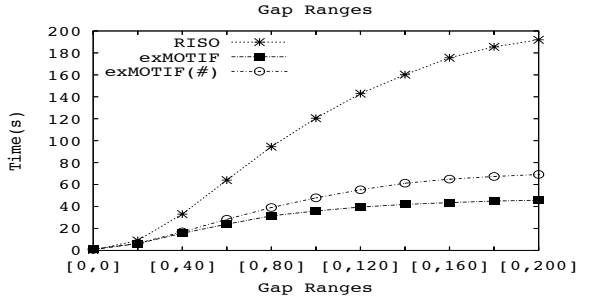
For comparison, we extract structured motifs from 1,062 non-coding sequences (a total of 196,736 nucleotides) located between two divergent genes in the genome of *B. subtilis* ([6; 7; 16]). Figure 3 and 4 compare the running time (in seconds) for exMotif and RISO using exact matching and approximate matching, respectively. Experiments were done for different gap ranges, number of components, and quorum thresholds. Note that exMotif has two options: one (shown as "exMOTIF" in the figures) for reporting only the number of sequences where the structured motifs occur, the other (shown as "exMOTIF(#)") for reporting both the number of sequences where the structured motifs occur and the actual occurrences. Also note that RISO *does not* report the actual occurrences; it reports only the frequency.



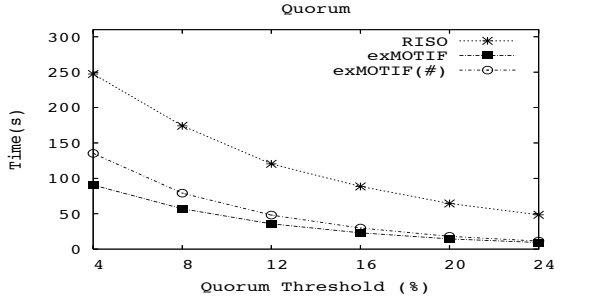
(a)



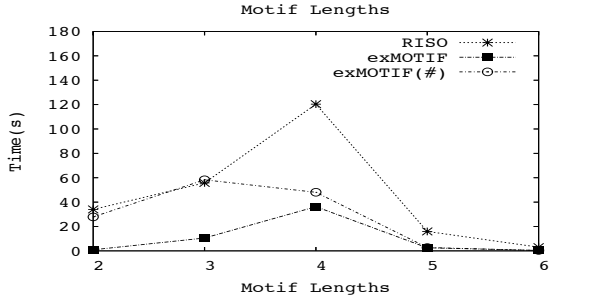
(b)



(c)

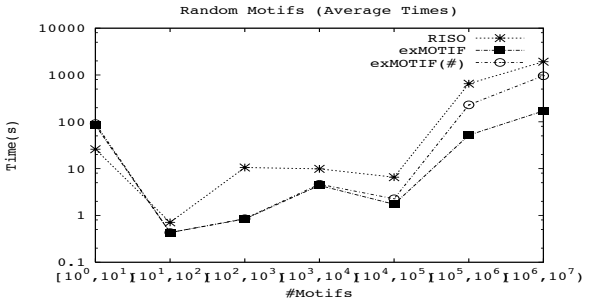


(d)

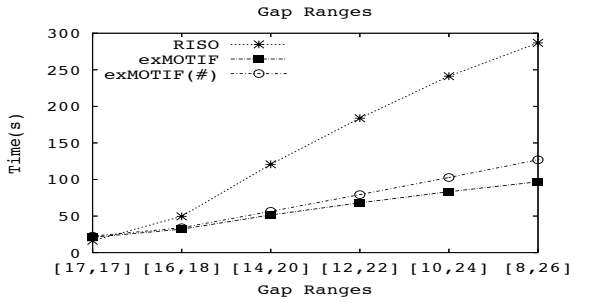


(e)

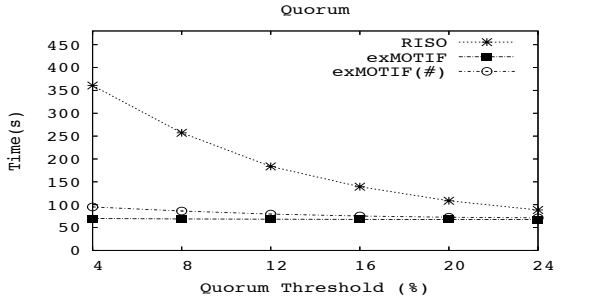
Figure 3: EXMOTIF vs. RISO: Exact Matching



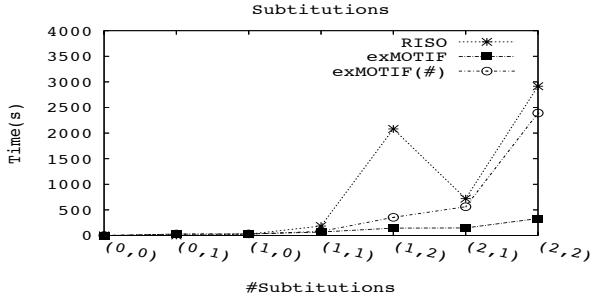
(a)



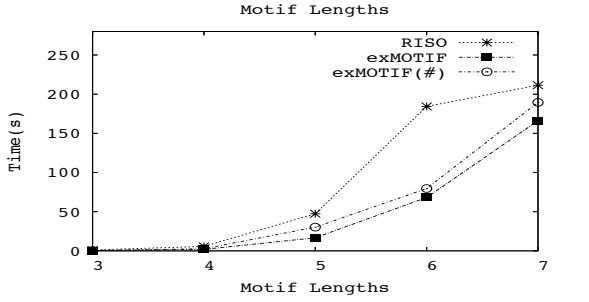
(b)



(c)



(d)



(e)

Figure 4: EXMOTIF vs. RISO: Approximate Matching

4.1.1 Exact Matching

In the first experiment, shown in Figure 3 (a), we randomly generated 100 structured motif templates, with $k \in [2,4]$ simple motifs of length $l \in [4,7]$ (k and l are selected uniformly at random within the given ranges). The gap range between each pair of simple motifs is a random sub-interval of $[0,200]$. The x-axis is sorted on the number of motifs extracted. For clarity we plot average times for the methods when the number of motifs extracted fall into the given range on the x-axis. For example, the time plotted for the range $[10^2, 10^3)$ is the average time for all the random templates that produce between 100 and 1000 motifs. We find that the average running time for RISO is 120.7s, whereas for exMotif it takes 88.4s for reporting only the support, and 91.3s for also reporting all the occurrences. The median times were 26.3s, 8.5s, and 9.2s, respectively, indicating a 3 times speed-up of exMotif over RISO.

In the next set of experiments we varied one parameter while keeping the others fixed. We set the default quorum to 12% ($q = 127$), the default gap ranges to [0,100], the default simple motif length to $l = 4$ (NNNN), and the default number of components $k = 3$ (e.g., NNNN[0,100]NNNN[0,100]NNNN). In Figure 3 (b), we plot the time as a function of the number of simple motifs k in the template. We find that as the number of components increases the time gap between exMotif and RISO increases; for $k = 4$ simple motifs, exMotif is around 5 times faster than RISO. Figure 3 (c) shows the effect of increasing gap ranges, from [0,0] to [0,200]. We find that as the gap range increases the time for exMotif increases at a slower rate compared to RISO. For [0,200], exMotif is 3-4 times faster than RISO depending whether only frequency or full occurrences are reported. In Figure 3 (d), as the quorum threshold increases, the running time goes down for both methods. For quorum 24%, exMotif is 4-5 times faster than RISO. As support decreases, the gap narrows somewhat, but exMotif remains 2-3 times faster. Finally, Figure 3 (e) plots the effect of increasing simple motif lengths $l \in [2,6]$. We find that the time first increases and then decreases. This is because there are a large number of motif occurrences for length 3 and length 4, but relatively few occurrences for length 5 and length 6. Depending on the motif lengths, exMotif can be 3-40 times faster than RISO for comparable output, i.e., reporting only the support. exMotif remains up to 5 times faster when also reporting the actual occurrences.

To compare the performance for extracting structured motifs with length ranges, we used the template $\mathcal{T} = M_1[50,100]M_2[1,50]M_3[20,100]M_4$ with $q = 12\%$, where $|M_1| \in [2,4]$, $|M_2| \in [3,4]$, $|M_3| \in [5,6]$, $|M_4| \in [4,5]$. exMotif took 78.4s, whereas RISO took 1640.9s to extract 14,174 motifs.

4.1.2 Approximate Matching

In the first experiment, shown in Figure 4 (a), we randomly generated 30 structured motif templates, with $k \in [2,3]$ simple motifs of length $l \in [3,6]$ (k and l are selected uniformly at random within the given ranges). The gap range between each pair of simple motifs is a random sub-interval of $[10,30]$. The x-axis is sorted on the number of motifs extracted, and average times are plotted for the extracted number of motifs in the given range. We find that the average running time for RISO is 334.5s, whereas for exMotif it takes 59.3s seconds for reporting only the support, and 176.7s for also reporting all the occurrences. Thus exMotif is on average 5 times faster than RISO, with comparable output.

Figures 4 (b)-(e) plot the time for approximate matching as a function of different parameters. We set the default quorum to 12% ($q = 127$, out of $|\mathcal{S}| = 1062$ sequences), the default gap ranges to [12,22], the default simple motif length to $l = 6$ (NNNNNN), and the default number of components $k = 2$ (e.g., NNNNNN[12,22]NNNNNN). Figure 4 (b) shows how increasing gap ranges effect the running time; for gap range [8,26] between the two motif components, exMotif is 2-3 times faster than RISO. In Figure 4 (c), we increase the numbers of arbitrary substitutions allowed for each simple motif; a pair (ϵ_1, ϵ_2) on the x-axis denotes that ϵ_1 substitutions are allowed for motif component M_1, and ϵ_2 for M_2. We can see that exMotif is always faster than RISO. It is 9 times faster when only frequencies are reported, and it can be up to 5 times faster then full occurrences are reported, though for some cases the difference is slight. Figure 4 (d) plots the effect of the quorum threshold. Compared to RISO, exMotif performs much better for low quorum, e.g., for $q = 4\%$ exMotif is 4-5 times faster than RISO. Finally in Figure 4 (e), as the simple motif lengths increase, the time for both exMotif and RISO increases, and we find that exMotif can be 2-3 times faster.

4.2 Real Applications

4.2.1 Discovery of Single Transcription Factor Binding Sites

We evaluate our algorithm by extracting the conserved features of known transcription factor binding sites in yeast. In particular we used the binding sites for the Zinc (Zn) factors ([22]). There are 11 binding sites listed for the Zn cluster, 3 of which are simple motifs. The remaining 8 are structured, as shown in Table 4. For the evaluation, we first form several structured motif templates according to the conserved features in the binding sites. Then we extract the frequent structured motifs satisfying these templates from the upstream regions of 68 genes regulated by zinc factors ([22]). We used the -1000 to -1 upstream regions, truncating the region if and where it overlaps with an upstream open-reading frame (ORF). After extraction, since binding sites cannot have many occurrences in the ORF regions, we drop some motifs if they also occur frequently in the ORF regions (i.e., within the genes). Finally, we calculate the Z-scores for the remaining frequent motifs, and rank them by descending Z-scores. In our experiments, we set the minimum quorum threshold to 7% within the upstream regions and the support threshold to 30% in the ORF regions. We use the shuffling program from SMILE ([12]) to compute the Z-scores. The shuffling program randomly shuffles the original input sequences to obtain a new *shuffled* set of sequences. Then it computes, for each extracted frequent motif, its support (π) and weighted support (π_w) in the shuffled set. For a given frequent motif $\mathcal{M}$, let μ and σ be the mean and standard deviation of its support across different sets (about 30) of shuffled sequences. Then the Z-score for each motif is calculated as: $\mathcal{Z} = \frac{\pi(\mathcal{M}) - \mu}{\sigma}$. Likewise we can also calculate the Z-score for each frequent motif by using the weighted support (which is also applicable for the repeated structured motif identification problem). As shown in Table 4, we can successfully predict GAL4, GAL4 chips, LEU3, PPR1 and PUT3 with the highest rank. CAT8 and LYS also have high ranks. We were thus able to extract all eight

Table 4: Regulons of Zn cluster proteins. `TF Name` stands for transcription factor name; `Known Motif` stands for the known binding sites corresponding to the transcription factors in `TF Name` column; `Predicted Motifs` stands for the motifs predicted by exMotif; `Num-Motifs` gives the final(original) number of motifs extracted (final is after pruning those motifs that are also frequent in the ORF regions); `Ranking` stands for the Z-score ranking based on support/weighted support.

TF Name	Known Motif	Predicted Motifs	Num-Motifs	Ranking
GAL4 GAL4 chips	CGGRnnRCYnYnCnCCG	CGG[11,11]CCG	1634(3346)	1/1
CAT8	CGGnnnnnnGGA	CGG[6,6]GGA	1621(3356)	147/13
HAP1	CGGnnnTAnCGG CGGnnnTAnCGGnnnTA	CGG[6,6]CGG	1621(3356)	111/146
LEU3	RCCGGnnCCGGY	CCG[4,4]CGG	1588(3366)	2/1
LYS	WWWTCCRnYGGAWWW	TCC[3,3]GGA	1605(3360)	33/21
PPR1	WYCGGnnWWYKCCGAW	CGG[6,6]CCG	1621(3356)	1/2
PUT3	YCGGnAnGCGnAnnnCCGA CGGnAnGCnAnnnCCGA	CGG[10,11]CCG	727(4035)	1/1

transcription factors for the Zinc factors with high confidence. As a comparison, with the same dataset RISO can only predict GAL4, LEU3 and PPR1.

4.2.2 Discovery of Composite Regulatory Patterns

The complex transcriptional regulatory network in Eukaryotic organisms usually requires interactions of multiple transcription factors. A potential application of exMotif is to extract such composite regulatory binding sites from DNA sequences. We took two such transcription factors, URS1H and UASH, which are involved in early meiotic expression during sporulation, and that are known to cooperatively regulate 11 yeast genes [21]. These 11 genes are also listed in SCPD [25], the promoter database of *Saccharomyces cerevisiae*. In 10 of those genes the URS1H binding site appears downstream from UASH; in the remaining one (HOP1) the binding sites are reversed. We took the binding sites for the 10 genes (all except HOP1), and after their multiple alignment, we obtained their consensus: taTTTtGGAGTaata[4,179]ttGGCGGCTAA (the lower case letters are less conserved, whereas uppercase letters are the most conserved). Table 5 shows the binding sites for UASH and URS1H for the 10 genes, their start positions, their alignment, and the consensus pattern. The gap between the sites are obtained after subtracting the length of UASH, 15, from the position difference (since the start position of UASH is given). The smallest gap is $l = 119 - 110 - 15 = 4$ and the largest is $u = 288 - 94 - 15 = 179$. Based on the on most conserved parts of the consensus, we formed the composite motif template: $\mathcal{T}$ = NNN[1,1]NNNNN[10,185]NNNNNNNNN (note the 6 additional gaps added to [4,179] to account for the non-conserved positions). We then extracted the structured motifs in the upstream regions of the 10 genes. We used the -800 to -1 upstream regions, and truncated the segment if it overlaps with an upstream ORF. The numbers of substitutions for NNN, NNNNN and NNNNNNNNN were set to $\epsilon_1 = 1$, $\epsilon_2 = 2$ and $\epsilon_3 = 1$, respectively. The quorum thresholds was set to $q = 0.7$ with the upstreams, and the maximum support within genes was set to 0.1% The rank of the true motif TTT[1,1]GGAGT[10,185]GGCGGCTAA was 290 (out of 5284 final motifs) with a Z-score of 22.61.

5. CONCLUSION AND FUTURE WORK

In this paper, we introduced exMotif, an efficient algorithm to extract structured motifs within one or multiple biological sequences. We showed its application in discovering single/composite regulatory binding sites. In the structured motif template, we assume the gap range between each pair of simple motifs is known. In the future, we plan to solve motif discovery problem when even the gap ranges are unknown. Another potential direction is to extract structured profile (or position weight matrix) patterns.

Acknowledgments

This work was supported in part by NSF CAREER Award IIS-0092978, DOE Career Award DE-FG02-02ER25538, and NSF grants EIA-0103708 & EMT-0432098.

6. REFERENCES

[1] Alberto Apostolico, Matteo Comin, and Laxmi Parida. Conservative extraction of over-represented extensible motifs. *Bioinformatics*, 21(Suppl. 1):i9–i18, 2005.

[2] Alberto Apostolico and Laxmi Parida. Incremental paradigms of motif discovery. *Journal of Computational Biology*, 11(1):15–25, 2004.

[3] Timothy L. Bailey and Charles Elkan. The value of prior knowledge in discovering motifs with MEME. In *3rd Int'l Conference on Intelligent Systems for Molecular Biology*, pages 21–29, 1995.

[4] Gary Benson. Tandem repeats finder: a program to analyze dna sequences. *Nucleic Acids Research*, 27(2):573–80, 1999.

[5] A. Brazma, I. Jonassen, J. Vilo, and E. Ukkonen. Pattern discovery in biosequences. In *International Colloquium on Grammatical Inference*, pages 257–270, 1998.

[6] A. Carvalho, A. Freitas, A. Oliveira, and M. Sagot. Efficient extraction of structured motifs using box-links. In *String Processing and Information Retrieval Conference*, pages 267–278, 2004.

Table 5: UASH and URS1H Binding Sites

Genes	UASH		URS1H		Gap
	Site	Pos	Site	Pos	
ZIP1	GATTCGGAAGTAAAA	-42	==TCGGCGGCTAAAT	-22	5
MEI4	TCTTTCGGAGTCATA	-121	==TGGGCGGCTAAAT	-98	8
DMC1	TTGTGTGGAGAGATA	-175	AAATAGCCGCCCA==	-143	17
SPO13	TAATTAGGAGTATAT	-119	AAATAGCCGCCGA==	-100	4
MER1	GGTTTTGTAGTTCTA	-152	TTTTAGCCGCCGA==	-115	22
SPO16	CATTGTGATGTATTT	-201	==TGGGCGGCTAAAA	-90	96
REC104	CAATTTGGAGTAGGC	-182	==TTGGCGGCTATTT	-93	74
RED1	ATTTCTGGAGATATC	-355	==TCAGCGGCTAAAT	-167	173
REC114	GATTTTGTAGGAATA	-288	==TGGGCGGCTAACT	-94	179
MEK1	TCATTTGTAGTTTAT	-233	==ATGGCGGCTAAAT	-150	68
Consensus	taTTTtGGAGTaata		==ttGGCGGCTAA==		[4,179]

[7] A. Carvalho, A. Freitas, A. Oliveira, and M. Sagot. A highly scalable algorithm for the extraction of cis-regulatory regions. In *Asia-Pacific Bioinformatics Conference*, pages 273–283, 2005.

[8] A. M. Carvalho, A. T. Freitas, A. L. Oliveira, and M.-F. Sagot. A parallel algorithm for the extraction of structured motifs. In *19th ACM Symposium on Applied Computing*, pages 147–153, 2004.

[9] E. Eskin, U. Keich, M.S. Gelfand, and P.A. Pevzner. Genome-wide analysis of bacterial promoter regions. In *The 8th Pacific Symposium on Biocomputing*, pages 29–40, 2003.

[10] E. Eskin and P.A. Pevzner. Finding composite regulatory patterns in dna sequences. *Bioinformatics*, 18 Suppl 1:S354–63, Jul 2002.

[11] Markus Friberg, Peter von Rohr, and Gaston Gonnet. Scoring functions for transcription factor binding site prediction. *BMC Bioinformatics*, 6(1):84, 2005.

[12] L. Marsan and M Sagot. Extracting structured motifs using a suffix tree - algorithms and application to promoter consensus identification. *Journal of Computational Biology*, 7:345–354, 2000.

[13] P. Michailidis and K. Margaritis. On-line approximate string searching algorithms: Survey and experimental results. *International Journal of Computer Mathematics*, 79(8):867–888, 2002.

[14] Giulio Pavesi, Giancarlo Mauri, and Graziano Pesole. An algorithm for finding signals of unknown length in dna sequences. *Bioinformatics*, 17(Suppl 1):S207–14, 2001.

[15] Giulio Pavesi, Giancarlo Mauri, and Graziano Pesole. A consensus based algorithm for finding transcription factor binding sites. In *Workshop on Genomes: Information, Structure, and Complexity*, 2004.

[16] N. Pisanti, A. M. Carvalho, L. Marsan, and M.-F. Sagot. Risotto: Fast extraction of motifs with mismatches. In *7th Latin American Theoretical Informatics Symposium*, 2006.

[17] A. Policriti, N. Vitacolonna, M. Morgante, and A. Zuccolo. Structured motifs search. In *Symposium on Research in Computational Molecular Biology*, pages 133–139, 2004.

[18] Marie-France Sagot. Spelling approximate repeated or common motifs using a suffix tree. In *3rd Latin American Symposium on Theoretical Informatics*, pages 374–390, 1998.

[19] Saurabh Sinha and Martin Tompa. Discovery of novel transcription factor binding sites by statistical overrepresentation. *Nucleic Acids Research*, 30(24):5549–60, 2002.

[20] Saurabh Sinha and Martin Tompa. YMF: a program for discovery of novel transcription factor binding sites by statistical overrepresentation. *Nucleic Acids Research*, 31(13):3586–3588, 2003.

[21] D.G. Thakurta and G.D. Stormo. Identifying target sites for cooperatively binding factors. *Bioinformatics*, 17(7):608–621, 2001.

[22] J. van Helden, A.F. Rios, and J. Collado-Vides. Discovering regulatory elements in non-coding sequences by analysis of spaced dyads. *Nucleic Acids Res*, 28(8):1808–18, Apr 2000.

[23] Mohammed J. Zaki. SPADE: An efficient algorithm for mining frequent sequences. *Machine Learning Journal*, 42:1–31, 2001.

[24] Minghua Zhang, Ben Kao, David Wai-Lok Cheung, and Kevin Yip. Mining periodic patterns with gap requirement from sequences. In *ACM Int'l Conference on Management of Data*, 2005.

[25] J. Zhu and M. Zhang. Scpd: A promoter database of the yeast saccharomyces cerevisiae. *Bioinformatics*, 15(7-8):607–11, 1999.

Motif Refinement using Hybrid Expectation Maximization based Neighborhood profile Search

Chandan K. Reddy
School of Electrical and Computer Engineering
Cornell University
Ithaca, NY - 14853

Yao-Chung Weng
School of Electrical and Computer Engineering
Cornell University
Ithaca, NY - 14853

Hsiao-Dong Chiang
School of Electrical and Computer Engineering
Cornell University
Ithaca, NY - 14853

ABSTRACT

The main goal of the motif finding problem is to detect novel, over-represented unknown signals in a set of sequences (for eg. transcription factor binding sites in a genome). Most widely used algorithms for finding motifs obtain a generative probabilistic representation of these over-represented signals and try to discover profiles that maximize the information content score. Although these profiles form a very powerful representation of the signals, the major difficulty arises from the fact that the best motif corresponds to the global maximum of a non-convex continuous function. Popular algorithms like Expectation Maximization (EM) and Gibbs sampling tend to be very sensitive to the initial guesses and are known to converge to the nearest local maximum very quickly. In order to improve the quality of the results, EM is used with multiple random starts or any other powerful stochastic global methods that might yield promising initial guesses (like projection algorithms). Global methods do not necessarily give initial guesses in the convergence region of the best local maximum but rather suggest that a promising solution is in the neighborhood region. In this paper, we introduce a novel optimization framework that searches the neighborhood regions of the initial alignments in a systematic manner to explore the multiple local optimal solutions. This effective search is achieved by transforming the original optimization problem into its corresponding dynamical system and estimating the practical stability boundary of the local maximum. Our results show that the popularly used EM algorithm often converges to sub-optimal solutions which can be significantly improved by the proposed neighborhood profile search. Based on experiments using both synthetic and real datasets, our method demonstrates significant improvements in the information content scores of the probabilistic models. The proposed method also gives the flexibility in using different local solvers and global methods that work well for some specific datasets.

Corresponding Author - Chandan K. Reddy - Email : ckr6@cornell.edu

General Terms

Bioinformatics, Data mining, Dynamical systems, Unsupervised learning.

Keywords

motif finding, global maximum, projection algorithms, expectation maximization, motif refinement

1. INTRODUCTION

Recent developments in DNA sequencing have allowed biologists to obtain complete genomes for several species. However, knowing the sequence does not imply the understanding of how these genes interact and regulate one another within the genome. Many transcription factor binding sites are usually highly conserved throughout the sequences and discovering the location of such binding sites plays an important role in the inference of the gene interaction and gene regulation. A *motif* is a sequence of DNA which manifests itself repetitively throughout genomic sequences. The length of the motif in each occurrence may not be the same as all of the other occurrences, although in general the occurrences must have roughly the same length. The motif challenge problem [19] that is being considered in this paper is described as follows: Given N sequences with t_i being the length of the i^{th} sequence, the goal of the motif finding problem is to locate all occurrences of the l-length motif which is within a distance of d mutations in each of the t sequences (see fig. 1). More details about the complexity of the motif finding problem is given in [18]. A detailed assessment of different motif finding algorithms has been published recently in [26].

GAATTCATACCAGATCAC**CGGATTCCCGA**CTCCAAATGTGTCCCCCTCACAC
TCCC**CCGAAAACCGA**CTTCTGCTCTTAGACCACTCTACCCTATTCCCCACACT
CACCGGAGCCAAAGCCGCGGCCCTTCCGTT**CCGATTACCGA**AAAGACCCCA
CCCGTAGGTGGCAAGCTAGCTTAAGTAACGCCACT**TCGATTAACGA**GGAAA
AATACATAACTGA**CCTATTATCGA**GTTCAGATCAAGGTCAGGAACAAAGAA
ACA**CCGATTACCGT**AACCGTAAGATARTGGTATCGATACGTAGACAGTTTA

Figure 1: Synthetic DNA sequences containing some instance of the pattern 'CCGATTACCGA' with a maximum number of 2 mutations. The motifs in each sequence are highlighted in the box. We have a (11,2) motif where 11 is the length of the motif and 2 is the number of mutations allowed.

Although there are several variations of the motif finding algorithms, the problem discussed in this paper is defined

as follows: without any previous knowledge about the consensus pattern, discover all instances (alignment positions) of the motifs and then recover the final pattern to which all these instances are within a given number of mutations. Inspite of the significant literature on the motif finding problem, relatively few researchers have exploited the probabilistic models used for motif refinement [16],[1]. More details on the estimates of the hardness of this problem without any complex information like overlapping motifs and background distribution is shown in [27].

In this paper, we provide a novel optimization framework for refining motifs using systematic subspace exploration and neighborhood search techniques. This paper is organized as follows: Section 2 gives some relevant background about the existing approaches used for finding motifs. Section 3 describes the problem formulation in detail. Section 4 discusses our new framework and Section 5 details our implementation. Section 6 gives the experimental results of our algorithm on synthetic and real datasets. Finally, Section 7 concludes our discussion with future research directions.

2. RELEVANT BACKGROUND

Existing approaches used to solve the motif finding problem can be classified into two main categories [10]. The first group of algorithms utilizes a generative probabilistic representation of the nucleotide positions to discover a consensus DNA pattern that maximizes information content score. In this approach, the original problem of finding the best consensus pattern is formulated into finding the global maximum of a continuous non-convex function. The main advantage of this approach is that profiles generated are highly representative of the signals being determined [7]. The disadvantage, however, is that the determination of the "best" motif cannot be guaranteed and is often very difficult since finding global maximum of any continuous non-convex function is a challenging problem. Current algorithms converge to the nearest local optimum instead of the global solution. Gibbs sampling [15], Expectation-Maximization [1], greedy CONSENSUS algorithm [13] and HMM based methods [8] belong to this category.

The second group uses patterns with 'mismatch representation' which defines a signal to be a consensus pattern and allows up to a certain number of mismatches to occur in each instance of the pattern. The goal of these algorithms is to recover the consensus pattern with the highest number of instances. These methods view the representation of the signals as discrete and the main advantage to these algorithms is that they can guarantee the highest scoring pattern to be the global optimum for any scoring function. The disadvantage, however, is that consensus patterns are not as expressive of the DNA signal as the profile representations. Recent approaches within this framework include Projection methods [4; 22], string based [19], Pattern-Branching [21], MULTIPROFILER [14], suffix trees [24] and other branch and bound approaches [11; 10].

A hybrid approach could potentially combine the expressiveness of the profile representation with convergence guarantees of the consensus pattern. An example of a hybrid approach is the Random Projection [4] algorithm followed by Expectation-Maximization [1]. It uses a global solver to obtain promising alignments in the discrete pattern space followed by further local solver refinements in continuous space[2; 25]. Currently, not many algorithms take complete advantage of the combined discrete and continuous space search [4; 10; 22]. In this paper, the profile representation of the motif is emphasize and a new hybrid algorithm is developed to escape out of the local maximum of the likelihood surface.

The main research concerns that motivated the new hybrid algorithm proposed in this paper are :

- Motif refinement stage is vital and popularly used by many pattern based algorithms (like PROJECTION, MITRA etc) that try to find optimal motifs.
- Traditional Expectation Maximization algorithm used in the context of motif finding converges very quickly to the nearest local optimal solution (within 5-8 iterations).
- There are many other promising local optimal solutions in the close vicinity of the profiles obtained from the global methods.

In spite of the importance of obtaining a globally optimal solution in the context of motif finding, not much work has been done in the direction of finding such solutions [28; 12]. There had been several attempts for escaping out of the local optimal solution to find better solutions in other machine learning [9] and optimization [5] related problems. Most of these methods are stochastic in nature and usually rely on perturbing either the data or the hypothesis. These stochastic perturbation algorithms are ineffecient because they sometimes miss a neighborhood solution or obtain an already existing solution. To avoid these problems, we introduce a novel optimization framework that has a better chance of avoiding sub-optimal solutions. It systematically escapes out of the convergence region of a local maximum to explore the existence of other neighborhood local maxima. Our method is primarily based on some fundamental principles of finding an exit points on the stability boundary of a nonlinear continuous function. The underlying theoretical details of our method are described in [6; 17].

3. PRELIMINARIES

Before discussing the details of our method, we describe our problem formulation and the details about the EM algorithm in the context of motif finding problem. We also describe some details about the dynamical system of the log-likelihood function which enables us to search the nearby local optimal solutions.

3.1 Problem Formulation

Some promising initial alignments are obtained by applying projection methods or random starts on the entire dataset. These initial alignments are then converted into profile representation.

Let t be the total number of sequences and n be the average length of each sequence. Let $S = \{S_1, S_2 ... S_t\}$ be the set of t sequences. Let $P = \{P_1, P_2 ... P_t\}$ be the set of initial alignments. l is the length of the consensus pattern. For further discussion, we use the following variables

$i = 1 \ldots t$ % for t sequences
$k = 1 \ldots l$ % for l-mers

Table 1: A count of nucleotides A, T, G, C at each position $K = 1..l$ in all the sequences of the data set. $K = 0$ denotes the background count.

j	$k=0$	$k=1$	$k=2$	$K=3$	$k=4$	...	$k=l$
A	$C_{0,1}$	$C_{1,1}$	$C_{2,1}$	$C_{3,1}$	$C_{4,1}$	...	$C_{l,1}$
T	$C_{0,2}$	$C_{1,2}$	$C_{2,2}$	$C_{3,2}$	$C_{4,2}$	...	$C_{l,2}$
G	$C_{0,3}$	$C_{1,3}$	$C_{2,3}$	$C_{3,3}$	$C_{4,3}$	...	$C_{l,3}$
C	$C_{0,4}$	$C_{1,4}$	$C_{2,4}$	$C_{3,4}$	$C_{4,4}$	...	$C_{l,4}$

$$j \in \{A, T, G, C\} \quad \% \text{ for each nucleotide}$$

The count matrix can be constructed from the given alignments as shown in Table 1. We define $C_{0,j}$ to be the non-position specific background count of each nucleotide in all of the sequences where $j \in \{A, T, C, G\}$ is the running total of nucleotides occurring in each of the l positions. Similarly, $C_{k,j}$ is the count of each nucleotide in the k^{th} position (of the $l - MER$) in all the P alignments.

$$Q_{0,j} = \frac{C_{0,j}}{\sum_j C_{0,j}} \tag{1}$$

$$Q_{k,j} = \frac{C_{k,j} + b_j}{t + \sum_j b_j} \tag{2}$$

Equation (1) shows the background frequency of each nucleotide where b_j is known as the Laplacian or Bayesian correction and is equal to $d * Q_{0,j}$ where d is some constant usually set to unity. Equation (2) gives the weight assigned to the type of nucleotide at the k^{th} position of the motif.
A Position Specific Scoring Matrix (PSSM) can be constructed from one set of instances in a given set of t sequences. From (1) and (2), it is obvious that the following relationship holds:

$$\sum_{j \in \{A,T,G,C\}} Q_{k,j} = 1 \qquad \forall k = 0, 1, 2, ...l \tag{3}$$

From equation (3), for a given k value, each Q can be represented in terms of the other 3 variables. Since the length of the motif is l, the final objective function (i.e. the information content score) would contain $3l$ independent variables[1]. To obtain the score, every possible $l - MER$ in each of the t sequences must be examined. This is done so by multiplying the respective $Q_{i,j}/Q_{0,j}$ dictated by the nucleotides and their respective positions within the $l - MER$. Only the highest scoring $l - MER$ in each sequence is noted and kept as part of the alignment. The total score is the sum of all the best scores in each sequence.

[1]Although, there are $4l$ variables in total, because of the constraints obtained from (3), the parameter space will contain only $3l$ independent variables. Thus, the constraints help in reducing the dimensionality of the search problem.

$$\begin{aligned} A(Q) &= \sum_{i=1}^{t} log(A)_i = \sum_{i=1}^{t} log\left(\prod_{k=1}^{l} \frac{Q_{k,j}}{Q_b}\right)_i \\ &= \sum_{i=1}^{t}\sum_{k=1}^{l} log(Q'_{k,j})_i \end{aligned} \tag{4}$$

$Q'_{k,j}$ is the ratio of the nucleotide probability to the corresponding background probability, i.e. $Q_{k,j}/Q_b$. $Log(A)_i$ is the score at each individual i^{th} sequence where t is the total number of sequences. In equation (4), we see that A is composed of the product of the weights for each individual position k. $A(Q)$ is the non-convex $3l$ dimensional continuous function for which the global maximum corresponds to the best possible motif in the dataset. EM refinement that is done at the end of the combinatorial approaches has the main disadvantage that it converges to a local optimal solution [3]. Our method improves the refinement procedure by understanding the details about the stability boundaries and trying to escape out of the convergence region of the EM algorithm.

3.2 Hessian Computation and Dynamical System for the Scoring Function

In order to present our algorithm, we have defined the dynamical system corresponding to the log-likelihood function and the PSSM. The key contribution of the paper is the development of this nonlinear dynamical system which will enable us to realize the dynamic and geometric nature of the likelihood surface. We construct the following *gradient system* in order to locate critical points of the objective function (4):

$$\dot{Q}(t) = -\nabla A(Q) \tag{5}$$

One can realize that this transformation preserves all the critical points [6]. Now, we will describe the construction of the gradient system and the Hessian in detail. In order to reduce the dominance of one variable over the other, the values of the each of the nucleotides that belong to the consensus pattern at the position k will be represented in terms of the other three nucleotides in that particular column. This will also minimize the dominance of the eigen vector directions when the Hessian is obtained. The variables of the scoring function are transformed into new variables described in Table 2.

$$A(Q) = \sum_{i=1}^{t}\sum_{k=1}^{l} log\ f_{ik}(w_{3k-2}, w_{3k-1}, w_{3k})_i \tag{6}$$

where

$$f_{ik} = \begin{cases} 1 - (w_{3k-2}, w_{3k-1}, w_{3k}) & \text{if } P_{ik} = C_k \\ w_{3k-2}\ or\ w_{3k-1}\ or\ w_{3k} & \text{elsewhere} \end{cases} \tag{7}$$

The first derivative of the scoring function is a one dimensional vector with $3l$ elements.

$$\nabla A = \left[\ \frac{\partial A}{\partial w_1}\ \ \frac{\partial A}{\partial w_2}\ \ \frac{\partial A}{\partial w_3}\ \ \ldots\ \ \frac{\partial A}{\partial w_{3l}}\ \right]^T \tag{8}$$

Table 2: A count of nucleotides $j \in \{A, T, G, C\}$ at each position $k = 1..l$ in all the sequences of the data set. C_k is the k^{th} nucleotide of the consensus pattern which represents the nucleotide with the highest value in that column. Let the consensus pattern be GACT...G and b_j indicates the background.

j	$k=b$	$k=1$	$k=2$	$K=3$	$k=4$	...	$k=l$
A	b_A	w_1	C_2	w_7	w_{10}	...	w_{3l-2}
T	b_T	w_2	w_4	w_8	C_4	...	w_{3l-1}
G	b_G	C_1	w_5	w_9	w_{11}	...	C_l
C	b_C	w_3	w_6	C_3	w_{12}	...	w_{3l}

and each partial derivative is given by

$$\frac{\partial A}{\partial w_p} = \sum_{i=1}^{t} \frac{\frac{\partial f_{ip}}{\partial w_p}}{f_{ik}(w_{3k-2}, w_{3k-1}, w_{3k})} \tag{9}$$

$$\forall p = 1, 2 \ ... \ 3l \quad and \quad k = round(p/3) + 1$$

The Hessian $\nabla^2 A$ is a block diagonal matrix of block size 3X3. For a given sequence, the entries of the 3X3 block will be the same if that nucleotide belongs to the consensus pattern (C_k). The gradient system is mainly obtained for enabling us to identify the stability boundaries and stability regions on the likelihood surface. The theoretical details about these concepts are published elsewhere [6]. Stability region of each local maximum is an approximate convergence zone of the EM algorithm. If we can identify all the saddle points on the stability boundary of a given local maximum, then we will be able to find all the tier-1 local maxima. However, finding all saddle points is computationally intractable and hence we have adopted a heuristic by generating the eigen vector directions of the PSSM at the local maximum. The next section details out our approach and explains the different phases of our algorithm.

4. NOVEL FRAMEWORK

Our framework consists of three phases. The first phase is the *global phase*, in which the promising solutions in the entire search space are obtained. The second phase is the *refinement phase* where a local method is applied to the solutions obtained in the previous phase in order to refine the profiles. The third phase is the *exit phase*; the exit points are computed and the Tier-1 and Tier-2 solutions are systematically explored.

In the global phase, a branch and bound search is performed on the entire dataset. All the profiles that do not meet a certain threshold (in terms of a given scoring function) are eliminated in this phase. The promising patterns obtained are transformed into profiles and local improvements are made to these profiles in the refinement phase. The consensus pattern is obtained from each nucleotide that corresponds to the largest value in each column of the PSSM. The $3l$ variables chosen are the nucleotides that correspond to those that are not present in the consensus pattern. Because of the probability constraints discussed in the previous section, the largest weight can be represented in terms of the other three variables.

To solve (4), current algorithms begin at random initial alignment positions and attempt to converge to an alignment of $l - MER$s in all the sequences that maximize the objective function. In other words, the $l - MER$ whose $log(A)_i$ is the highest (with a given PSSM) is noted in every sequence as part of the current alignment. During the maximization of $A(Q)$ function, the probability weight matrix and hence the corresponding alignments of $l - MER$s are updated. This will occur iteratively until the PSSM converges to the locally optimal solution. The consensus pattern is obtained from the largest weight nucleotide in each position (column) of the PSSM. This converged PSSM and the set of alignments correspond to a local optimal solution. To escape out of this local optimal solution, our approach requires the computation of a Hessian matrix (i.e. the matrix of second derivatives) of dimension $(3l)^2$ and the $3l$ eigenvectors of the Hessian. The Hessian $\nabla^2 A$ is a block diagonal

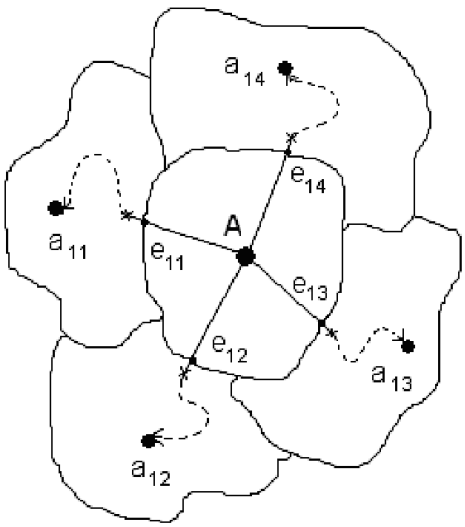


Figure 2: Diagram illustrates the exit point method of escaping from the original solution (A) to the neighborhood local optimal solutions (a_{1i}) through the corresponding exit points (e_{1i}). The dotted lines indicate the local convergence of the EM algorithm.

Input: Local Maximum (A).
Output: Best Local Maximum in the neighborhood region.
Algorithm:
Step 1: Construct the PSSM for the alignments corresponding to the local maximum (A) using Eqs. 1 and 2.
Step 2: Calculate the eigen vectors of the Hessian matrix for this PSSM.
Step 3: Find exit points (e_{1i}) on the practical stability boundary along each eigen vector direction.
Step 4: For each of the exit points, the corresponding Tier-1 local maxima are obtained (a_{1i}) by applying the EM algorithm after the ascent step.
Step 5: Repeat this process for promising tier-1 solutions to obtain Tier-2 (a_{2j}) local maxima.
Step 6: Return the solution that gives the maximum information content score of $\{A,\ a_{1i},\ a_{2j}\}$.

Figure 3: The Exit phase where the neighborhood of the original solution is explored in a systematic manner.

matrix of block size 3X3. For a given sequence, the entries of the 3X3 block will be the same if that nucleotide belongs to the consensus pattern (C_k). The main reasons for choosing the eigenvectors of the Hessian as search directions are:

- Computing eigen vectors of the Hessian is related to finding the directions with extreme values of the second derivatives, i.e., directions of extreme normaltoisosurface change.
- Eigen vectors of the Hessian will form the basis vectors for the search directions. Any other search direction can be obtained by a linear combination of these basis directions.
- This will make our algorithm deterministic since the eigen vector directions are always unique.

The value of the objective function is evaluated along these eigen vector directions with some small step size increments. Since the starting position is a local optimal solution of a function that is being maximized, the function value during the initial few steps will reduce. Since the Hessian is obtained only once during the entire procedure, it is more efficient compared to Newton's method where an approximate Hessian is obtained for every iteration. After a certain number of step evaluations, there might be an increase in the value indicating that the current point is out of the convergence region of the local maximum. The point along this direction where the $A(Q)$ has the lowest value is called the *exit point*. Once the exit points are computed along each eigenvector direction, the local maximum in the other region is obtained by applying local method with these new points as initial guesses. This procedure is clearly shown in Fig 4. To ascertain that the new initial guess is in a different convergence region from the original, the objective function value is evaluated even after its increase. The descent stage indicates the function evaluation along a particular eigen vector direction. Applying local method at the exit point might give the original local maximum. The ascent stage is used to ensure that the new guess is in a different convergence zone. Hence, given the best local maximum obtained using any current local methods, this framework allows us to systematically escape out of the local maximum to explore surrounding local maxima.

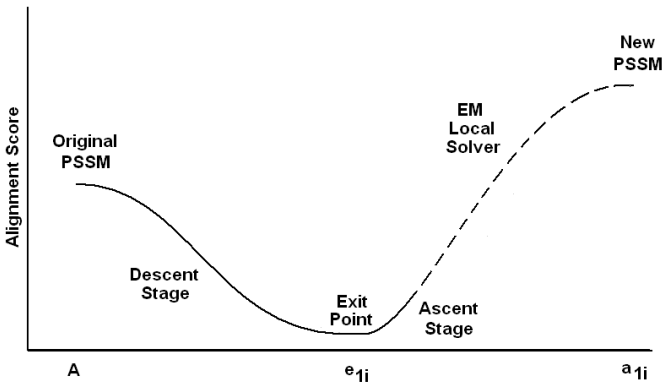


Figure 4: A summary of escaping out of the local optimum to the neighborhood local optimum. Observe the corresponding trend of $A(Q)$ at each step.

Input: The DNA sequences, length of the motif (l), Maximum Number of Mutations (d)
Output: Motif (s)
Algorithm:
Step 1: Given the sequences, apply random projection algorithm to obtain different set of alignments.
Step 2: Choose the promising buckets and apply EM algorithm to refine these alignments.
Step 3: Apply the exit point method to obtain nearby promising local optimal solutions.
Step 4: Report the consensus pattern that corresponds to the best alignments and their corresponding PSSM.

Figure 5: The complete algorithm

This new framework can be treated as a hybrid approach between global method and the local method. The approach differs from traditional local methods by computing multiple local solutions in the neighborhood region in a systematic manner. It differs from global methods by working completely in profile space and searching a subspace efficiently in a deterministic manner. For a given non-convex function, there is a massive number of convergence regions that are very close to each other and are separated from one another in the form of different basins of attraction. These basins are effectively modeled by the concepts of stability regions.

5. IMPLEMENTATION DETAILS

Our program is implemented in Red Hat Linux version 9 and runs on a Pentium IV 2.8 GHz machine. The core algorithm that we implemented is XP_EM described in Algorithm 1. XP_EM obtains the initial alignments and the original data sequences along with the length of the motif. It returns the best motif that is obtained in the neighboring region of the sequences. This procedure constructs the PSSM, performs EM refinement, and then computes the Tier-1 and Tier-2 solutions by calling the procedure $Next\_Tier$. The Eigen vectors of the Hessian were computed using the code obtained from [20]. $Next\_Tier$ takes a PSSM as input and computes an array of PSSMs corresponding to the next tier local maxima using the exit point methodology.

Algorithm 1 Motif $XP\_EM(init\_aligns, seqs, l)$

$PSSM = Construct\_PSSM(init\_aligns)$
$New\_PSSM = Apply\_EM(PSSM, seqs)$
$TIER1 = Next\_Tier(seqs, New\_PSSM, l)$
for $i = 1$ to $3l$ **do**
  if $TIER1[i] <> zeros(4l)$ **then**
    $TIER2[i][\,] = Next\_Tier(seqs, TIER1[i], l)$
  end if
end for
Return $best(PSSM, TIER1, TIER2)$

Given a set of initial alignments, Algorithm 1 will find the best possible motif in the neighborhood space of the profiles. Initially, a PSSM is computed using *construct_PSSM* from the given alignments. The procedure *Apply_EM* will return a new PSSM that corresponds to the alignments obtained after the Expectation Maximization algorithm is applied to the initial PSSM. The details of the procedure $Next\_Tier$

are given in Algorithm 2. From a given local solution (or PSSM), $Next\_Tier$ will compute all the $3l$ new $PSSM$s in the neighborhood of the given local optimal solution. The second tier patterns are obtained by calling the $Next\_Tier$ from every first tier solutions [2]. Finally, the pattern with the highest score amongst the original PSSM, Tier1 and Tier2 is returned.

Algorithm 2 PSSMs[ ] $Next\_Tier(seqs, PSSM, l)$

```
Score = eval(PSSM)
Hess = Construct_Hessian(PSSM)
Eig[ ] = Compute_EigVec(Hess)
MAX_Iter = 100
for k = 1 to 3l do
  PSSMs[k] = PSSM        Count = 0
  Old_Score = Score      ep_reached = FALSE
  while (! ep_reached) && (Count < MAX_Iter) do
    PSSMs[k] = update(PSSMs[k], Eig[k], step)
    Count = Count + 1
    New_Score = eval(PSSMs[k])
    if (New_Score > Old_Score) then
      ep_reached = TRUE
    end if
    Old_Score = New_Score
  end while
  if count < MAX_Iter then
    PSSMs[k] = update(PSSMs[k], Eig[k], ASC)
    PSSMs[k] = Apply_EM(PSSMs[k], Seqs)
  else
    PSSMs[k] = zeros(4l)
  end if
end for
Return PSSMs[ ]
```

The procedure $Next\_Tier$ takes a PSSM and computes an array of PSSMs that corresponds to the next tier local optimal solutions. It applies the Exit-point methodology to compute the next tier solution. The procedure *eval* evaluates the scoring function for the PSSM using (4). The procedures $Construct\_Hessian$ and $Compute\_EigVec$ computes the Hessian matrix and the eigen vectors respectively. MAX_iter indicates the maximum number of uphill evaluations that are required along each of the eigen vector directions. The neighborhood PSSMs will be stored in the variable $PSSMs[\ ]$. The original PSSM is updated with a small step until an exit point is reached or the number of iterations exceed MAX_Iter value. If the exit point is reached along a particular direction, some more iterations are made to guarantee that the PSSM exists in a different stability region and entered a new one. The EM algorithm is then used during this ascent stage to obtain a new PSSM [3].

The initial alignments are converted into profile space and a PSSM is constructed. The PSSM is updated (using the EM algorithm) until the alignments converge to a local optimal solution. The Exit-point methodology is then employed to escape out of this local optimal solution to compute nearby first tier local optimal solutions. This process is then repeated on promising first tier solutions to obtain second tier solutions. As shown in Fig. 6, from the original local optimal solution, various exit points and correspondingly the new local optimal solutions are computed along each Eigen vector direction. Sometimes two directions might yield the same local optimal solution. This can be avoided by computing the saddle point corresponding to the exit point on the stability boundary [23]. There can be many exit points, but there will be a unique saddle point corresponding to the new local minimum. However, in high dimensional problems, it is not very efficient to compute the saddle points and hence, we chose to compute the exit points. For computational efficiency, the Exit-point approach is applied to only promising initial alignments (i.e. random starts with higher Information Content score). Therefore, a threshold $A(Q)$ score is determined by the average of the best three first tier scores just after 10-15 random starts; any current and future first tier solution with score greater than the threshold is considered for further analysis. Additional random starts are carried out in order to aggregate at least ten first tier solutions. Exit-point is repeated on all first tier solutions above a certain threshold to obtain second tier solutions.

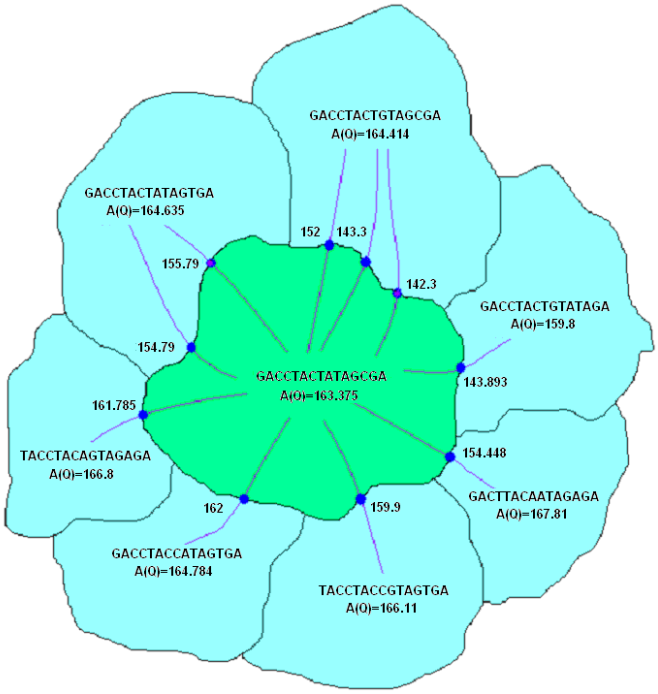


Figure 6: 2-D illustration of first tier improvements in a $3l$ dimensional objective function. The original local maximum has a score of 163.375. The various Tier-1 solutions and plotted and the one with highest score (167.81) is chosen.

6. EXPERIMENTAL RESULTS

Experiments were performed on both synthetic data and real data. Two different methods were used in the global phase:

[2]New PSSMs might not be obtained for certain search directions. In those cases, a zero vector of length $4l$ is returned back. Only those new PSSMs which do not have this value will be used for any further processing.

[3]For completeness, the entire algorithm has been shown in this section. However, during the implementation, several heuristics have been applied to reduce the running time of the algorithm. For example, if the first tier solution is not very promising, it will not be considered for obtaining the corresponding second tier solutions.

random starts and random projection. The main purpose of this paper is not to demonstrate that our algorithm can outperform the existing motif finding algorithms. Rather, the main work here focusses on improving the results that are obtained from other efficient algorithms. We have chosen to demonstrate the performance of our algorithm on the results obtained from the random projection method which is a powerful global method that has outperformed other traditional motif finding approaches like MEME, Gibbs sampling, WINNOWER, SP-STAR etc [4]. Since the comparison results were already published, we mainly focus on the performance improvements of our algorithm compared to the random projection algorithm. For random starts experiment, a total of N random numbers each value between 1 and $(t - l + 1)$ that corresponds to random initial starting alignments are generated. Let m be the number of independent trials required to obtain the motifs.

6.1 Synthetic Datasets

The synthetic datasets were generated using the procedure described in [19]. The value of $m = 1$ is chosen to demonstrate the efficiency of our approach. This corresponds to one full random projection + EM cycle. We compared the performance coefficient (PC) which gives the measure of the average performance of our implementation to that of Random Projection. The PC is given by :

$$PC = \frac{|K \cap P|}{|K \cup P|} \tag{10}$$

where K is the set of the residue positions of the planted motif instances, and P is the corresponding set of positions predicted by the algorithm. Table 4 gives an overview of the performance of our method compared to the random projection algorithm on the (l,d) motif problem for different l and d values.

Our results also show that by branching out and discovering multiple local optimal solutions one need not use higher m values. A higher m value corresponds to more computational time because projecting the l-mers into k-sized buckets is a time consuming task. Using our approach, we can replace the need for randomly projecting l-mers repeatedly in an effort to converge to a global optimum by deterministically and systematically searching the solution space modeled by our dynamical system and improving the quality of the existing solutions. The improvements of our algorithm are clearly shown in Table 4. We can see that there is a significant improvement for higher length motifs.

Table 4: The results of performance coefficient with $m = 1$ on synthetically generated sequences. The scores are not normalized and the perfect score is 20 since there are 20 sequences.

Motif (l,d)	PC obtained using Random Projection	PC obtained using Exit-point method
(11,2)	20	20
(15,4)	14.875	17
(20,6)	12.667	18

Fig. 6 shows the tier-1 solutions obtained from a given consensus pattern. Since the exit points are being used instead of saddle points, it might sometimes find the same local optimal solution obtained before. As seen from the figure, the tier-1 solutions does not have to be different from the original pattern in just one nucleotide position. Also, the function value at the exit points is much higher than the original value.

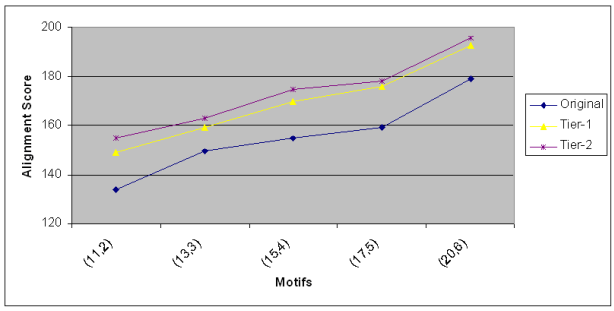


Figure 7: The average scores with the corresponding first tier and second tier improvements on synthetic data using the random starts with Exit-point approach with different (l,d) motifs.

As opposed to stochastic processes like mutations in genetic algorithms, our approach reduces the stochastic nature and tries to obtain multiple local optimal solutions in the neighborhood systematically. Fig. 7 shows the performance of the Exit-point approach on synthetic data for different (l,d) motifs. The average scores of the best ten solutions obtained from random starts and their corresponding improvements in tier-1 and tier-2 are reported. One can see that the improvements become more prominent for larger length motifs. Table 3 shows the best and worst of these top ten random starts along with the consensus pattern and the alignment scores.

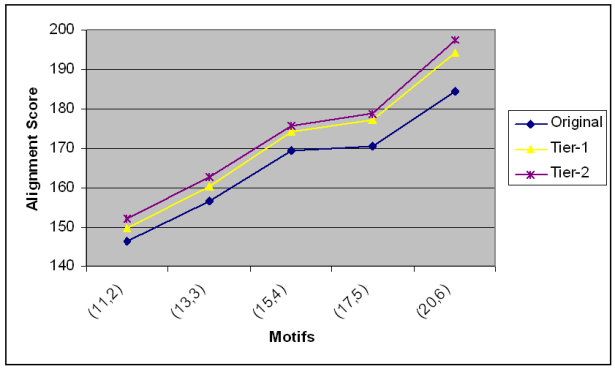


Figure 8: The average scores with the corresponding first tier and second tier improvements on synthetic data using the Random Projection with Exit-point approach with different (l,d) motifs.

With a few modifications, more experiments were conducted using the Random Projection method. The Random Projection will eliminate non-promising regions in the search space and gives a number of promising sets of initial patterns. EM refinement is applied to those promising initial patterns with higher score. Due to the robustness of the results, the Exit-point method is employed only on the top five local optima. Exit-point is again repeated on the top scoring first tier so-

Table 3: The consensus patterns and their corresponding scores of the original local optimal solution obtained from multiple random starts on the synthetics data. The best first tier and the second tier optimal patterns and their corresponding scores are also reported.

(l,d)	Initial Pattern	Score	First Tier Pattern	Score	Second Tier Pattern	Score
(11,2)	AACGGTCGCAG	125.1	CCCGGTCGCTG	147.1	CCCGGGAGCTG	153.3
(11,2)	ATACCAGTTAC	145.7	ATACCAGTTTC	151.3	ATACCAGGGTC	153.6
(13,3)	CTACGGTCGTCTT	142.6	CCACGGTTGTCTC	157.8	CCTCGGGTTTGTC	158.7
(13,3)	GACGCTAGGGGGT	158.3	GAGGCTGGGCAGT	161.7	GACCTTGGGTATT	165.8
(15,4)	CCGAAAAGAGTCCGA	147.5	CCGCAATGACTGGGT	169.1	CCGAAAGGACTGCGT	176.2
(15,4)	TGGGTGATGCCTATG	164.6	TGGGTGATGCCTATG	166.7	TGAGAGATGCCTATG	170.4
(17,5)	TTGTAGCAAAGGCTAAA	143.3	CAGTAGCAAAGACTACC	173.3	CAGTAGCAAAGACTTCC	175.8
(17,5)	ATCGCGAAAGGTTGTGG	174.1	ATCGCGAAAGGATGTGG	176.7	ATTGCGAAAGAATGTGG	178.3
(20,6)	CTGGTGATTGAGATCATCAT	165.9	CAGATGGTTGAGATCACCTT	186.9	CATTTAGCTGAGTTCACCTT	194.9
(20,6)	GGTCACTTAGTGGCGCCATG	216.3	GGTCACTTAGTGGCGCCATG	218.8	CGTCACTTAGTCGCGCCATG	219.7

lutions to arrive at the second tier solutions. Fig. 8 shows the average alignment scores of the best random projection alignments and their corresponding improvements in tier-1 and tier-2 are reported. In general, the improvement in the first tier solution is more significant than the improvements in the second tier solutions.

6.2 Real Datasets

Table 5 shows the results of the Exit-point methodology on real biological sequences. We have chosen $l = 20$ and $d = 2$. 't' indicates the number of sequences in the real data. The m value reported is the approximate average number of full cycles required to obtain the motif. For the biological samples taken from [4; 21], the value of m is the average number of full LSH cycles it would take the original algorithm to discover the motif. The values for all other parameters (like projection size $k = 7$ and threshold s=4) are chosen to be the same as those used in the Random projection paper [4]. All the motifs were recovered with $m = 1$ using the Exit-point strategy. Without the exit point strategy, the random projection algorithm needed multiple LSH cycles inorder to retrieve the original motifs. This clearly elucidates the fact that, we need to use global methods to certain extent and combine them with refined local heuristics in order to obtain better efficiency. Running one LSH cycle is much more time consuming compared to the exit-point strategy. The main advantage of our strategy comes with the deterministic nature of the algorithm as opposed to the stochastic version (as seen in random projection). This clearly indicates the efficiency of the newly proposed method on real biological samples.

7. CONCLUDING DISCUSSION

The Exit-point framework proposed in this paper broadens the search region for obtaining improved solution that can potentially corresponds to a better motif. In most of the profile based algorithms, EM is used to obtain the nearest local optimum from a given starting point. In our approach, we consider the boundaries of these convergence regions and find the surrounding local optimal solution based on the theory of stability regions. We have shown in both real and synthetic data sets that beginning from the EM converged solution, the Exit-point approach is capable of searching in the neighborhood regions for another solution with an improved information content score. This will often translate to finding a pattern with less hamming distance from the resulting alignments in each sequence. Our approach has shown improvements in the score on all datasets that it was tested on. One of the primary advantages of our method is that it can be used with different global and local methods. The main contribution of our work is to demonstrate the capability of this hybrid expectation maximization algorithm in the context of the motif finding problem. We can potentially use any global method and improve its results efficiently.

From our results, we see that motif refinement stage in the motif finding problem plays a vital role and can yield accurate results more efficiently in terms of computational costs. We would like to continue our work by combining other global methods that are available in the literature with existing local solvers like EM or GibbsDNA that work in continuous space. Implementing the Exit-point method as an intermediate between the global and local solver provides us with a fundamental advantage of choosing different methods to explore the data specific properties in more detail. We will follow the example of [26] and try different combinations of the existing methods to improve the chances of finding more promising patterns.

8. REFERENCES

[1] T. Bailey and C. Elkan. Fitting a mixture model by expectation maximization to discover motifs in biopolymers. *The First International Conference on Intelligent Systems for Molecular Biology*, pages 28–36, 1994.

[2] Y. Barash, G. Bejerano, and N. Friedman. A simple hyper-geometric approach for discovering putative transcription factor binding sites. *Proc. of First International Workshop on Algorithms in Bioinformatics*, 2001.

[3] K. Blekas, D. Fotiadis, and A. Likas. Greedy mixture learning for multiple motif discovery in biological sequences. *Bioinformatics*, 19(5):607–617, 2003.

[4] J. Buhler and M. Tompa. Finding motifs using random projections. *Proceedings of the fifth annual international conference on Research in computational molecular biology*, pages 69–76, 2001.

[5] B. C. Cetin, J. Barhen, and J. W. Burdick. Terminal repeller unconstrained subenergy tunneling (TRUST) for fast global optimization. *Journal of Optimization Theory and Applications*, 77(1):97–126, 1993.

[6] H.D. Chiang and C.C. Chu. A systematic search method for obtaining multiple local optimal solutions of

Table 5: Results of Exit-point methodology on biological samples. The real motifs were obtained in all the six cases using the Exit-point framework.

Sequence	Sample Size	t	Best (20,2) Motif	Reference Motif
E. coli CRP	1890	18	TGTGAAATAGATCACATTTT	TGTGANNNNGNTCACA
preproinsulin	7689	4	GGAAATTGCAGCCTCAGCCC	CCTCAGCCC
DHFR	800	4	CTGCAATTTCGCGCCAAACT	ATTTCNNGCCA
metallothionein	6823	4	CCCTCTGCGCCCGGACCGGT	TGCRCYCGG
c-fos	3695	5	CCATATTAGGACATCTGCGT	CCATATTAGAGACTCT
yeast ECB	5000	5	GTATTTCCCGTTTAGGAAAA	TTTCCCNNTNAGGAAA

nonlinear programming problems. *IEEE Transactions on Circuits and Systems: I Fundamental Theory and Applications*, 43(2):99–109, 1996.

[7] R. Durbin, S.R. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis : Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1999.

[8] S. R. Eddy. Profile hidden markov models. *Bioinformatics*, 14(9):755–763, 1998.

[9] G. Elidan, M. Ninio, N. Friedman, and D. Schuurmans. Data perturbation for escaping local maxima in learning. *In Proceedings of the Eighteenth National Conference on Artificial Intelligence*, pages 132 – 139, 2002.

[10] E. Eskin. From profiles to patterns and back again: A branch and bound algorithm for finding near optimal motif profiles. *Proceedings of the eighth annual international conference on Research in computational molecular biology*, pages 115–124, 2004.

[11] E. Eskin and P. Pevzner. Finding composite regulatory patterns in dna sequences. *Tenth International Conference on Intelligent Systems for Molecular Biology*, pages 354–363, 2002.

[12] G. B. Fogel, D. G. Weekes, G. Varga, E. R. Dow, H. B. Harlow, J. E. Onyia, and C. Su1. Discovery of sequence motifs related to coexpression of genes using evolutionary computation. *Nucleic Acids Research*, 32(13):3826–3835, 2004.

[13] G. Hertz and G. Stormo. Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics*, 15(7/8):563577, 1999.

[14] U. Keich and P. Pevzner. Finding motifs in the twilight zone. *Bioinformatics*, 18:1374–1381, 2002.

[15] C. Lawrence, S. Altschul, M. Boguski, J. Liu, A. Neuwald, and J. Wootton. Detecting subtule sequence signals: a gibbs sampling strategy for multiple alignment. *Science*, 262:208–214, 1993.

[16] C. E. Lawrence and A. A. Reilly. An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences. *Proteins: Structure, Function, and Genetics*, 7:41–51, 1990.

[17] J. Lee and H.D. Chiang. A dynamical trajectory-based methodology for systematically computing multiple optimal solutions of general nonlinear programming problems. *IEEE Transactions on Automatic Control*, 49(6):888 – 899, 2004.

[18] P. Pevzner. *Computational Molecular Biology - an algorithmic approach*, chapter Finding Signals in DNA, pages 133–152. MIT Press, 2000.

[19] P. Pevzner and S-H. Sze. Combinatorial approaches to finding subtle signals in DNA sequences. *The Eighth International Conference on Intelligent Systems for Molecular Biology*, pages 269–278, 2000.

[20] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1992.

[21] A. Price, S. Ramabhadran, and P.A. Pevzner. Finding subtle motifs by branching from sample strings. *Bioinformatics*, 1(1):1–7, 2003.

[22] B. Raphael, L.T. Liu, and G. Varghese. A uniform projection method for motif discovery in DNA sequences. *IEEE Transactions on Computational biology and Bioinformatics*, 1(2):91–94, 2004.

[23] C. K. Reddy and H.D. Chiang. A stability boundary based method for finding saddle points on potential energy surfaces. *Journal of Computational Biology*, 13(3):745–766, 2006.

[24] M. Sagot. Spelling approximate or repeated motifs using a suffix tree. *Lecture Notes in Computer Science*, 1380:111–127, 1998.

[25] E. Segal, Y. Barash, I. Simon, N. Friedman, and D. Koller. From promoter sequence to expression: a probabilistic framework. In *Proceedings of the sixth annual international conference on Computational biology*, pages 263 – 272, Washington, DC, USA, 2002.

[26] M. Tompa and N. Li et al. Assessing computational tools for the discovery of transcription factor binding sites. *Nature Biotechnology*, 23(1):137 – 144, 2005.

[27] M. Waterman, R. Arratia, and E. Galas. Pattern recognition in several sequences: consensus and alignment. *Mathematical Biology*, 46:515–527, 1984.

[28] E. Xing, W. Wu, M.I. Jordan, and R. Karp. LOGOS: A modular bayesian model for de novo motif detection. *Journal of Bioinformatics and Computational Biology*, 2(1):127–154, 2004.

Protein Classification using Summaries of Profile-Based Frequency Matrices

Keith Marsolo
Department of Computer Science and Engineering
The Ohio State University
Columbus, OH, USA

Srinivasan Parthasarathy
Department of Computer Science and Engineering
The Ohio State University
Columbus, OH, USA
srini@cse.ohio-state.edu

ABSTRACT

The ability to detect or predict the structural class of a protein based on its primary sequence has been a major objective for researchers working in bioinformatics. Within the bioinformatics community, the prevailing belief seems to be that support vector machines (SVMs) are the most effective solution for sequence-based structure prediction. The current state-of-the-art involves SVMs that employ kernel functions designed to compute the similarity between proteins based on profiles generated by the PSI-BLAST alignment algorithm. While effective for problems such as fold recognition or remote homology detection, these kernels are essentially a "black-box" solution to the structure prediction problem. They do not yield a representation that is independent of the SVM. This prevents the user from testing alternative classification algorithms or from using the features for other applications.

For example, there may be instances where a researcher is interested in a compact representation of a protein sequence that can be used for problems such as range queries or nearest-neighbor retrieval. We describe such a representation in this work. Using the frequency scores returned by PSI-BLAST, we create a wavelet-based summary. This stand-alone, normalized feature vector drastically reduces the amount of information that needs to be stored for each protein. Though our results are preliminary, empirically, we find that this representation performs well in both experiments dealing with fold recognition and provides accuracy comparable to the state-of-the-art for remote homology detection. At the same time, we find that it is also effective for protein indexing and retrieval.

Keywords

Bioinformatics, sequence-based representation, protein classification

1. INTRODUCTION

The past few years has seen an explosion of sequence data, both genomic and proteomic. Unlike genes, proteins have a functional role that is influenced by their structure. Solving the structure of a protein is not as easy or straightforward as determining its sequence, however. Rather than wait for the structure of every protein to be solved, researchers have looked to predict structure based on an input sequence. A large number of approaches exist, but more popular methods include Hidden Markov Models (HMMs) [14], profile-based alignment methods such as PSI-BLAST [1] and methods based on support vector machines (SVMs) [6; 12; 23; 10; 27; 17; 16; 15]. Each method has its proponents, but a majority seems to be leaning toward SVM-based approaches as being the most effective form of structure prediction.

The more successful SVM-based approaches incorporate a specialized kernel function into the learning process. Kernel functions are designed to compute a high-dimensional similarity between objects in the dataset. A hyperplane can then be constructed in this high-dimensional feature space, partitioning the object space. These kernel functions have shown to be quite effective, but they can be computationally expensive. While some users might be willing to trade speed for accuracy, there is another drawback to these kernels in that they are essentially "black-box" functions. There is no intermediate representation that one can use for other machine learning or data mining problems. The closest thing to an independent feature vector is a pair-wise matrix that illustrates the relationship between every object in the dataset. As the size of the dataset increases, any attempts at manipulating this matrix or using it as a representation quickly become infeasible.

Thus, if one was interested in tasks *other* than SVM-based classification, it is imperative to have a stand-alone representation. Such feature vectors could be used by alternative classification algorithms or in applications such as nearest-neighbor or range-based similarity search. Of course, it would also be desirable if this representation could also be used for SVM-based structure prediction while providing results that are comparable to the state-of-the-art.

In this paper, we present such a representation. Like many existing approaches to structure prediction, it is derived from the results of multiple sequence alignments. Using wavelets, we derive a summary from the alignment profile that results in a compact, stand-alone feature vector that can be used in multiple applications. In an empirical study, we test the performance of our approach on a number of protein datasets. We conduct experiments in SVM-based classification and prediction and evaluate the potential use of our summary in alternative applications such as nearest neighbor-based similarity search. Though our results are still preliminary, it appears that our method provides excel-

lent accuracy on the problem of fold recognition, and comparable performance to the state-of-the-art for the problem of remote homology detection. We also find that it holds promise for use in protein indexing and similarity-based retrieval. In the sections that follow, we describe the construction of our summary and provide experimental results that illustrate its effectiveness.

2. BACKGROUND

Proteins are comprised of a varying sequence of 20 different amino acids. These amino acids combine to form interacting subunits called secondary structures. The interactions between secondary structures give a protein its overall shape, which is often called the protein's *fold*. Several structural databases have arisen to group proteins based on their fold. One of the most popular is the Structural Classification of Proteins (SCOP) Database[20].

SCOP arranges proteins into several hierarchical levels. The first four are *Class*, *Fold*, *SuperFamily* and *Family*. Proteins in the same Class share similar secondary structure information, while proteins within the same Fold have similar secondary structures that are arranged in the same topological configuration. Proteins in the same SuperFamily show clear structural homology and proteins belonging to the same Family exhibit a great deal of sequence similarity and are thought to be evolutionarily related.

The SCOP hierarchy is useful when validating any structural classification/prediction algorithm. This validation usually occurs in one of two ways. The first is through the process of fold recognition. With fold recognition, an algorithm is tested by trying to classify proteins at a particular level of the hierarchy, traditionally the Fold or SuperFamily level[1]. In these experiments, the training and testing data contain members of all the possible folds. Thus, such tests can be viewed as similar to traditional classification. Remote homology detection, on the other hand, is closer to structure prediction. In these tests, entire Families (or possibly SuperFamilies) are held out of the training data. A learning model is trained and then tested to see whether it can accurately predict the labels of the test data one level "up" the hierarchy. In other words, if a Family is held out as test data, a predictor will be considered correct if can accurately identify the SuperFamily of the held-out Family (or Fold for SuperFamily). This is often considered to be the more challenging of the two validation methods.

PSI-BLAST

PSI-BLAST is one of the more popular methods used to determine the similarity of a protein against a database of sequences. This similarity is returned in the form of a profile, or scoring matrix. In PSI-BLAST, a sequence is tested against a database to identify conserved patterns, or motifs. For each position in each of these conserved regions, the algorithm computes a score for each amino acid type. In highly conserved regions, those amino acids that are highly conserved receive a high positive score, while the others receive high negatives. In weakly conserved regions, residues receive scores near zero. These scores are calculated based on amino acid frequency information. Evolution-based substitution matrices such as BLOSUM [11] can also be used when calculating the scores. This process can be iterative, running a profile against the database to refine the results. The final output of the algorithm is a profile that includes a position-specific scoring matrix (PSSM) and the position-specific amino acid frequency matrix (PSFM), as well as a sequence of Z-scores or E-values that denote the statistical significance of the alignment.

[1]We use the uppercase *Fold* when referring to the SCOP classification. The lowercase *fold* is used to denote a generic group of structurally-similar proteins.

SVM Kernel Functions

In an attempt to improve prediction accuracy over traditional SVM-based approaches, many researchers have looked at incorporating kernel functions into the learning process. These kernel functions have specific mathematical properties, but their main purpose is to calculate a high-dimensional similarity between pairs of objects based on a particular representation of the input data. This similarity is determined by calculating the inner (dot) product between the objects. Once the high-dimensional similarities for all objects have been computed, a hyperplane is constructed to partition the object space [3]. We will now briefly review a few of the kernel functions that have been created to solve the structure prediction problem.

The *mismatch string kernel* represents two input sequences A and B as a series of subsequences of size k. For a fixed number of mismatches, the kernel will compute the number of subsequences of sequence A that exist in list of subsequences of sequence B [17]. With the *spectrum kernel*, for all k-length subsequences, the kernel computes the frequency of occurrence, or in binary form, simply the presence of a subsequence, and uses that information to calculate similarity [16]. *Profile kernels*, are similar to mismatch kernels, but use probabilistic alignment scoring matrices to define a positional mutation neighborhood, which are used to determine similarity [15]. A second variation of the profile kernel computes similarity based on both the scoring and frequency matrices [23]. Groups have also looked at combining profile kernels with *adaptive codes*. Here, SVMs are trained using profile kernels and the results are used to to train a perceptron that provides a final prediction [12]. Profile kernels have also been used with *cluster kernels* for semi-supervised learning. Cluster kernels rely on profile kernels to create a clustering and this information is used to classify labeled and unlabeled proteins [27]. Finally, the Fisher-SVM was created to calculate similarity between sequences based on Fisher scores that were derived from a profile-based HMM [13].

3. APPROACH

In this section we detail the steps needed to create the wavelet-based profile summary. We begin with a discussion on the creation of the PSI-BLAST profile. We follow with a brief overview of the wavelet decomposition and conclude with a description of the process used to compute our summary.

Profile Generation

The first step in creating our representation involves generating a PSI-BLAST profile for each protein. Using version 2.2.13 of the algorithm, we compute a multi-way alignment against the non-redundant (*nr*) protein database. Downloaded in March 2006, this database contains almost 3.5

million sequences. We ran PSI-BLAST for five iterations with the ϵ parameter set to 0.001. Since our representation is derived from the position-specific frequency matrix (PSFM), we set the program to output both that and the position-specific scoring matrix (PSSM), in addition to the standard alignment information. We provide a graphical illustration of the matrices for protein 1HL2A in Figure 1. Each row corresponds to an amino acid, while each column represents a position in the query sequence. These images are false color representations, so lower values have darker colors, while higher values appear lighter. Once the profile has been generated, we create a summary of the frequency information by applying a 1D wavelet decomposition to the PSFM matrix.

Wavelet-Based Compression

The use of wavelets is natural in applications that require a high degree of compression without a corresponding loss of detail, or where the detection of subtle distortions and discontinuities is crucial [18]. Wavelet transformations fall into two separate categories: continuous (*cwt*) and discrete (*dwt*). Here, we deal with the discrete wavelet transform. Given a one-dimensional signal of length N (typically a power of two), the *dwt* consists of at most $\log N$ stages. At each stage, two sets of coefficients are produced, the approximation and detail. The approximation coefficients are generated by convolving the original signal with a low-pass filter, the detail with a high-pass filter. After passing the signal through the filter, the results are downsampled by a factor of two. This step is repeated on the approximation coefficients, producing another smaller set of approximation and detail coefficients. There are a number of different wavelet families and each of these families has a corresponding low-pass and high-pass filter. In our tests, we focus on the Haar wavelet, which is the simplest of the wavelet families. In the past, we have experimented with other families, but found that for our purposes, there is little difference among their performance.

Profile Normalization

Before we can apply the wavelet decomposition, however, we must normalize the size of the matrix to ensure that we are left with the same number of coefficients for each protein. This will allow us freely compare between the resulting feature vectors. The PSFM and PSSM matrices are of size nx20, where n refers to the number of amino acids in the protein sequence and 20 corresponds to one of the different amino acid types. We fix n to be 128 and normalize each PSFM matrix to 128x20. The normalization occurs either through interpolation or extrapolation, depending on whether the input protein is shorter or longer than 128 residues, respectively. We choose a value of 128 because wavelet transformations are most effective on signals whose length is a power of 2. We prefer to interpolate and smooth or average the excess points, over extrapolation, where we would be forced to generate additional data. Most of the proteins in our datasets are shorter than 256 residues, thus our choice of 128. The normalization is computed using the Matlab 'interp2' command.

Once the matrix has been normalized we transpose it (20 rows x 128 columns) and apply a 5th level Haar 1D decomposition to each row. We only use the final level of approximation coefficients, so a decomposition will produce 1 coefficient for every 32 input values (2^5), or 4 coefficients per row/amino acid. This results in a feature vector of size 80, which we use when conducting our experiments. We provide a flowchart illustrating the transformation process in Figure 3. An example of the wavelet-based representations can be seen in Figure 2. The PSSM representation is given on the left (Figure 2 (a)), while the PSFM version lies on the right (Figure 2 (b)).

Transformation Details

The wavelet decomposition creates a frequency signature for each amino acid. Using a 5th level Haar decomposition results in 4 coefficients. The Haar wavelet filter is an averaging filter, so each of these coefficients will represent the average frequency value for 25% of the sequence, multiplied by a filter-based scaling and normalization factor. One could vary the decomposition or the size of the normalized profile to change the number of coefficients per amino acid, which would change the representation percentage, but a tradeoff must be made between the total number of coefficients and the overall accuracy of the representation. For instance, a 4th level decomposition will result in a total of 160 coefficients/protein, while a 6th level yields just 40. We tried several different parameter settings but found that a 5th level decomposition gave us the best results.

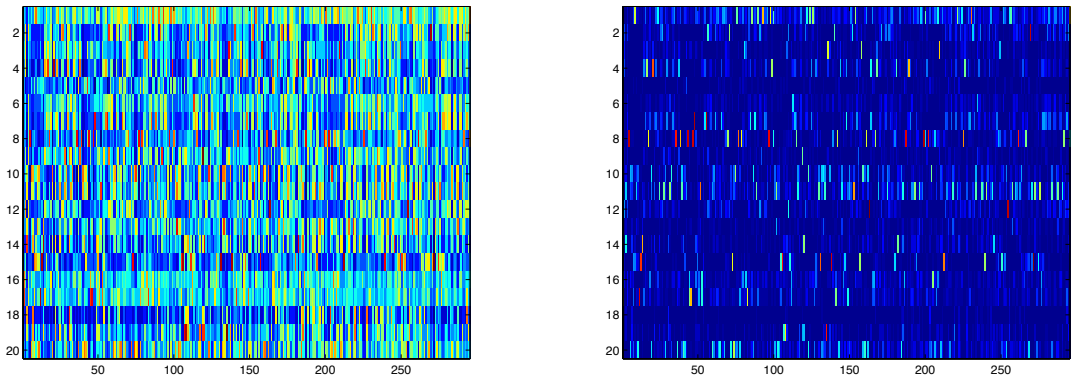
Since the Haar wavelet filter is orthonormal, each coefficient represents a non-overlapping segment of the protein sequence. If one wanted to create a representation that includes such an overlap, this could be done by manually calculating the average using a sliding window, or through other techniques such as n-grams. If n were set to 32, one would start at position one, and then compute an n-gram for the first 32 points, slide to the right one position and compute a second n-gram. This process would repeat down the profile. Such an approach would increase the total number of coefficients, however. We use wavelets instead of manually computing the summary because of their speed and relative ease of implementation.

We also tested our approach on the scoring matrix, which is traditionally used by profile kernels, but found that we achieved better overall results using the raw frequency values. We detail this information in our experimental results. We denote the frequency-based vectors as *PSFM* and the score-based vectors as *PSSM*. We surmise that the PSFM summary outperforms the PSSM-based measure because the entries in the scoring matrix represent log-odd ratios, which scales the data and removes some of the variance that may help distinguish between the different groups. As an example, for one of our datasets, all of the PSSM values fall between -7 and 13. The PSFM matrices, on the other hand, contain values between 0 and 100. The wider variance of the PSFM matrix is preferable to the tighter range of the PSSM values. This is analogous choosing the covariance matrix over correlation for EM-based missing value analysis [21].

4. DATASETS

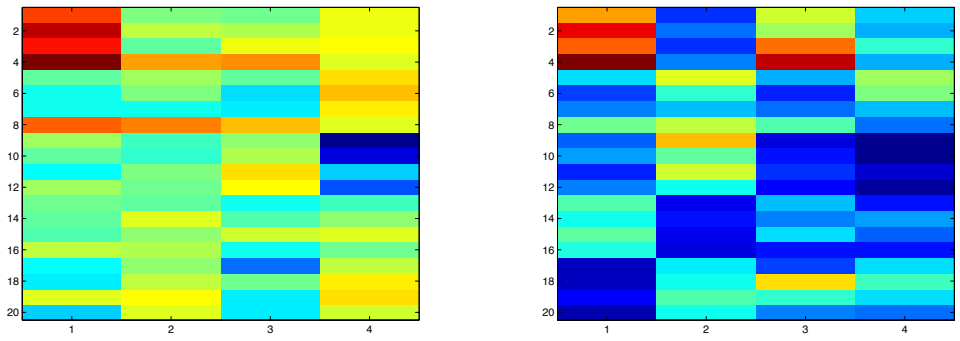
In an attempt to compare with some of the existing approaches, we run experiments on multiple datasets derived from the SCOP database. A number of variations have been reported in the literature, each of which makes the task of classification more or less challenging. Thus, we note the dataset parameters when describing each experiment.

First, we examine a dataset originally derived by Ding and



(a) (b)

Figure 1: Graphical illustration of the PSI-BLAST profile for protein 1HL2A. The image on the left (a) represents the PSSM matrix, while the one on the right (b) corresponds to the PSFM values. The figures provide a false color representation of the matrices, with lower values having a darker color while higher values appear lighter.



(a) (b)

Figure 2: Graphical illustration of the wavelet-based profile summaries for protein 1HL2A. The image on the left (a) represents the PSSM representation, while the one on the right (b) corresponds to the PSFM-based version. The figures provide a false color representation of the matrices, with lower values having a darker color while higher values appear lighter.

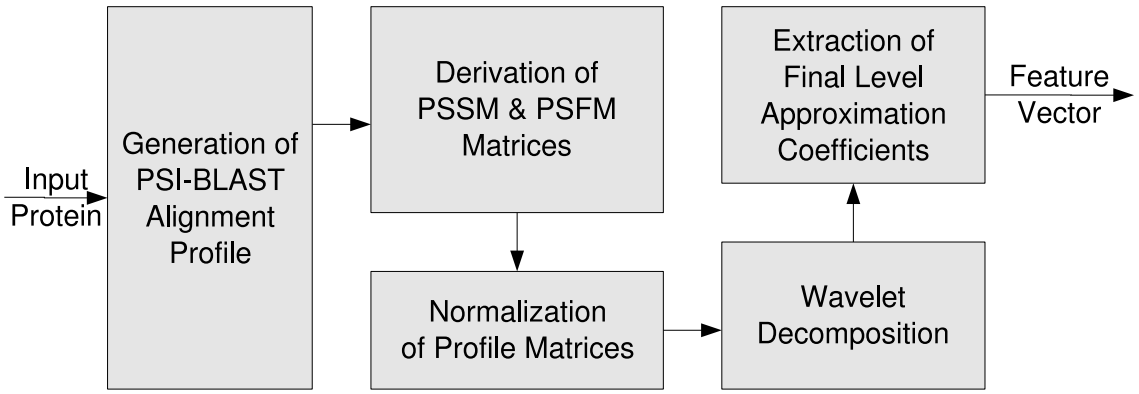


Figure 3: Flowchart of the proposed transformation.

Dubchak [8], but later studied in a number of publications [26; 7; 25; 19; 12; 24]. Here, a training set was taken from the 27 most populated SCOP folds of the PDB_Select set, in which no two proteins share more than 35% sequence identity for aligned subsequences longer than 80 residues. This training set contained 311 proteins. An independent test set was derived from the PDB_40D set, which consists of all SCOP sequences having less than 40% sequence identity with each other. Using the same 27 SCOP folds, 385 proteins were selected, and any PDB_40D protein having more than 35% sequence identity with the proteins in the training set was excluded. When combined together, the training and test sets yield a total of 696 proteins. Since the publication of the work by Ding and Dubchak [8], the protein identifiers used in the SCOP database have changed. We were looking to match the original identifiers to those currently used by SCOP but where unable to do so for 46 proteins. Thus, our version of this dataset, which we call the *Ding & Dubchak* set, contains 653 proteins.

We next examine a dataset was used by Ie et al. in their multi-class SVM study [12]. In that work, they took proteins from version 1.65 of the SCOP database and used ASTRAL [4] to filter the proteins so no two members shared more than 95% sequence identity. They then removed all SCOP Folds with less than three SuperFamilies. This resulted in a dataset containing 24 Folds and 46 SuperFamilies. In addition, they further held out 46 SCOP Families to use for remote homology detection. These Families comprise less than 40% of the total sequences in their corresponding SuperFamilies, leaving 60% of that SuperFamily available for training. The held-out Families were not used in any part of the training process. In the end, they were left with 1772 sequences in the training set and 338 in the held-out test set. Our attempts to replicate this dataset resulted in 1611 and 310 sequences, respectively. We refer to this dataset as the *24Fold* set.

Starting with the 24Fold dataset, we also examined a subset where the proteins contained less than 40% sequence identity. Our efforts here were an attempt to replicate those experiments dealing with more remotely-related proteins [6; 10]. After removing the proteins that shared more than 40% identity, we are left with training and testing splits of 667 and 185 proteins, respectively. We refer to both of these datasets as the *24Fold* sets but provide the percentage of sequence identity when discussing our results.

The final dataset has been examined in a number of database indexing experiments [2; 5], most recently by Gao and Zaki [9]. Again taken from the SCOP database (though a different version than the one used in constructing the 24Fold sets), this set contains a total of 1810 proteins taken from 181 different SuperFamilies containing at least 10 proteins, though only 10 proteins were selected per SuperFamily. Of the 1810 proteins, we could not match 22 proteins, leaving us with a total of 1798. We label this variant as *181SF*.

5. EXPERIMENTS

All tests were conducted on a 2.4 GHz Pentium 4 PC with 1.5 GB RAM running Ubuntu Linux on a 2.6.12 kernel. For our classification experiments, we use the SVM provided by WEKA, version 3.4 [28]. WEKA uses Platt's sequential minimal optimization (SMO) algorithm for SVM training [22]. We use a linear kernel with the default complexity parameter (1.0) and train each classifier in a one-vs-rest fashion. We report classification performance in terms of the average accuracy, with accuracy values reported as the number of true positives divided by the number of true positives plus the number of true negatives (TP/(TP+TN)), returned as a percentage. plus the number of true negatives, returned as a percentage.

Representation	Accuracy
CSHPVZ	57.2
PSSM	59.7
PSFM	64.0

Table 1: SVM accuracy of a standard sequence-based representation versus the PSSM and PSFM-based summaries on the Ding & Dubchak dataset

PSFM Summaries vs. Sequence-based Representation

Our first experiment is an attempt to compare our approach against a stand-alone representation that is derived from a number of sequence-based properties. These features were first described in Ding and Dubchak [8], though they were used in a number of subsequent experiments by other research groups [8; 26; 7; 25]. The feature vectors characterize the following properties for each protein (the symbol for each descriptor given in parentheses): amino acid composition (c), hydrophobicity (h), polarity (p), predicted secondary structure (s), van der Waals volume (v), and polarizability (z). The feature vector for the amino acid composition consists of 20 dimensions. All of the rest have 21. When combined, each protein is represented by a total of 125 features.

Taking the *Ding & Dubchak* dataset, we train an SVM with a linear kernel using 10-fold cross-validation. We compare the sequence-based properties used by Ding and Dubchak versus our PSFM and PSSM-based summaries. We report these accuracy values in Table 1. As we can see, the PSFM-based features outperform the combined feature vector used by Ding and Dubchak. It also outperforms or is comparable to results reported in a number of previous publications [26; 7; 25; 24]. In addition, the PSFM-based summaries result in an accuracy that is almost 5% higher than the PSSM-based values.

PSFM Summaries vs. Multiple Profile-based Properties

Our second set of experiments are an attempt to compare against results reported by Cheng and Baldi and their approach, which they call FOLDpro [6]. In that work, the similarity between two proteins was calculated using 54 different properties including alignments from CLUSTALW, PSI-BLAST, COMPASS, a number of sequence-based properties, structural information and more (the similarity between PSI-BLAST profiles is considered to be just one property in their feature vector). Taking a dataset of 976 proteins sharing less than 40% sequence identity, an SVM was trained on these similarity measures using 10-fold cross-validation. Each fold required 3 days of training time. Using the top 1 and top 5 results to assign a final label, Cheng and Baldi looked at the accuracy of their method at the Fold, Super-

Method	Fold	SuperFamily	Family
PSFM-1NN	74.3	70.7	69.0
FOLDpro	26.5	55.5	85.0

Table 2: Results for a 1-NN classifier against those reported by FOLDpro

Sequence Identity	Features	Fold Recognition	Remote Homology Detection
$< 40\%$	PSSM	77.5	58.9
$< 40\%$	PSFM	74.7	70.2
$< 95\%$	PSSM	90.5	63.8
$< 95\%$	PSFM	88.2	72.3

Table 3: Accuracy of Fold Recognition and Remote Homology Detection experiments for the PSSM and PSFM summaries on the 24Fold datasets.

Family and Family level. We compare their SVM results against a simple 1-NN classifier, trained using 10-fold cross-validation, which requires just minutes to construct.
Though the datasets are not exactly the same, and we have a lower accuracy at the Family level, we do have a higher accuracy at other levels in the SCOP hierarchy. We are 50% higher at the Fold level, and 15% higher at the SuperFamily level. In addition, these results are achieved with a simple 1-NN classifier that computes the similarity between our PSFM-based feature vectors. The similarity between PSI-BLAST profiles is just one of the the 54 properties that Cheng and Baldi use in their classifier, however.

Fold Recognition & Remote Homology Detection

Next, we run a series of tests designed to replicate those used to evaluate the performance of the spectrum, mismatch, and profile kernels [12; 27; 17; 16; 15]. Here we perform two different sets of experiments. The first, which we view as Fold Recognition, involves training an SVM using 10-fold cross validation. Though we are classifying proteins at the Fold level, members of each underlying Family exist in both the training and testing splits. The second set of tests is designed to detect homologous proteins that have low sequence similarity. In these experiments, which we refer to as Remote Homology Detection, the test split consists of 46 Families that are completely held out of the training phase, providing a totally independent test set. These experiments follow those reported elsewhere [12]. We report results for the datasets containing less than 40% sequence identity as well as the set with less than 95% identity. There is no consensus within the bioinformatics community as to which is preferable, so we report results on both.
Table 3 lists the results for our Fold Recognition and Remote Homology Detection experiments. Unlike the Ding & Dubchak results, in the task of Fold Recognition, the PSSM summary is comparable to the PSFM version. The results also show the effect that sequence identity has on the results. As one might expect, having a higher sequence identity results in a more accurate classifier. The last column of Table 3 shows the results of our remote homology experiments. Though we use labels from the SuperFamily level, the classifier has not been trained on the Families in our test set. Therefore, these experiments will be a true test of whether our approach captures the hierarchical relationship that exists between Families belonging to the same SuperFamily. This poses a much greater challenge, and as one might expect, we see accuracy values that are a bit lower than the fold recognition experiments.
Unlike the fold recognition experiments, however, the PSFM representation provides a higher accuracy than PSSM. The PSSM results drop by 20-30%, compared to 4-15% with the PSFM representation. While there is a decrease with both identity datasets, the decrease on the dataset with less than 95% identity is much larger than the dataset with less than 40% identity. Thus, one could argue that the PSFM representation is truly effective at detecting remote homologies since the dataset with less than 40% identity represents a much more challenging classification problem. Regardless, we find that our representation provides comparable results to the performance of a one-vs-all SVM trained with a profile kernel, which returned an accuracy of around 78% on this dataset [12]. While the use of certain optimizations or the training of a second-level classifier can boost performance, for a single SVM, we feel our approach is more than adequate.

Sequence Identity	Kernel	Fold Recognition	Remote Homology Detection
$< 40\%$	RBF	31.0	18.4
$< 40\%$	Linear	74.7	70.2
$< 95\%$	RBF	57.0	53.0
$< 95\%$	Linear	88.2	72.3

Table 4: Accuracy of Fold Recognition and Remote Homology Detection experiments for the PSFM representation on the 24Fold dataset using an SVM trained with a RBF and linear kernel.

Effect of Kernel on Accuracy

Our final SVM-based experiments were designed to examine the effect of kernel type on accuracy. Here we focus on two of the more popular stock SVM kernels: linear and radial basis function (RBF). For approaches that do not implement a custom kernel, it has been reported that RBF kernels are superior to linear [6]. We repeat the fold recognition and remote homology experiments described above using the PSFM representation. As we see in Table 4, the linear kernel vastly outperforms the RBF kernel for our PSFM-based summary. RBF kernels compute similarity based on the difference between objects, while linear kernels rely on the inner product. We surmise that since the difference between objects is likely to be small, it is affecting the classification process.

KNN-based Similarity Search

Finally, we compare against a leading structural indexing algorithm, PSIST [9], to see how our PSFM-based summary

Dataset	Class	Fold	SuperFamily	Family
PSIST	98.3	−−	93.9	−−
PSFM	98.3	98.3	97.8	96.3

Table 5: 3-NN classification accuracy of PSFM-based summary versus results returned by PSIST.

functions as a database index. If our representation is to be used in other applications, it should be comparable to existing work. In PSIST, a feature vector is generated for each protein based on the distances and angles between residues. These vectors are placed into a suffix-tree, which serves as the indexing structure to retrieve similar proteins, given a query. We obtained the source code for PSIST[2] and tested a number of different parameter settings, but found that the default parameters provided the best results.

We examine the predictive accuracy of our PSFM representation by using a k-d tree as a nearest-neighbor classifier. Using the 181SF dataset, we select 1 protein from each of the 181 SuperFamilies to serve as a query protein. Given a query, we retrieve its three nearest-neighbors from the the target database. We then compare the SCOP labels of the query and the retrieved proteins. If the labels of at least two of the neighbors match, we say that the query has been classified correctly (the query proteins are not included in the target database). Similar classification experiments are detailed in the paper on PSIST [9]. In that work, proteins were only compared at the Class and SuperFamily level. We also examine the accuracy of our method at the Fold and Family levels. As one progresses down the SCOP hierarchy, the protein structures become increasingly similar and the distinctions between them more fine-grained. A truly useful representation should not lose accuracy during this descent, even though the classification problem itself becomes more difficult.

As one can see in Table 5 our method provides excellent results, providing comparable or higher accuracy than PSIST at the provided Class and SuperFamily levels. In addition, the accuracy of our PSFM representation drops only slightly as one progresses down the hierarchy. This implies that the proteins we are retrieving are truly correct and that our representation has potential for use elsewhere. While PSIST is limited to solved structures, our approach can be applied to all sequenced proteins.

6. CONCLUSIONS AND ONGOING WORK

In this paper, we present an approach that can be employed to generate a stand-alone representation of a protein sequence using the results of the PSI-BLAST alignment algorithm. We construct a summary of the PSI-BLAST frequency matrix using wavelets and use this as our representation. We explore the possibility of using a score-based summary, but find its performance to be inferior to that of the frequency-based summary. We feel this difference is the result of the log-based scaling that is applied to the scoring matrix. We evaluate our summary on a number of protein datasets by conducting SVM-based prediction and classification experiments. We find that our summary provides comparable accuracy to state-of-the-art kernel-based SVMs with the potential for use in other areas. This added capability is demonstrated by the performance of our representation against a leading structural indexing technique using nearest neighbor-based similarity search. Though we conduct an empirical study, and provide preliminary results, we feel that such a representation will be useful for protein researchers. As part of ongoing work, we plan a more robust and rigorous evaluation as well as an exploration of other potential applications for its use.

[2]We would like to thank F. Gao and M. Zaki for providing the source code and for their assistance in interpreting the results.

7. REFERENCES

[1] S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Anang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucleic Acids Research*, 25:3389–3402, 1997.

[2] A. Bhattacharya, T. Can, T. Kahveci, A. Singh, and Y. Wang. ProGreSS: Simultaneous searching of protein databases by sequence and structure. In *Pacific Symposium on Biocomputing*, volume 9, pages 264–275. World Scientific Press, 2004.

[3] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *5th Annual ACM Workshop on COLT*, pages 144–152, Pittsburgh, PA, 1992. ACM Press.

[4] S. E. Brenner, P. Koehl, and M. Levitt. The ASTRAL compendium for sequence and structure analysis. *Nucleic Acids Research*, pages 254–256, 2000.

[5] O. Çamoğlu, T. Kahveci, and A. Singh. Towards index-based similarity search for protein structure databases. In *2nd IEEE Computer Society Bioinformatics Conference (CSB)*, pages 148–158, 2003.

[6] J. Cheng and P. Baldi. A machine learning information retrieval approach to protein fold recognition. *Bioinformatics*, 2006.

[7] A. Chinnasamy, W. K. Sung, and A. Mittal. Protein structure and fold prediction using tree-augmented naive bayesian classifier. In *Proc. PSB 2004*, Stanford, CA, 2004. World Scientific Press.

[8] C. H. Q. Ding and I. Dubchak. Multi-class protein fold recognition using support vector machines and neural networks. *Bioinformatics*, 17(4):349–358, April 2001.

[9] F. Gao and M. Zaki. PSIST: Indexing protein structures using suffix trees. In *IEEE Computational Systems Bioinformatics Conference*, Palo Alto, CA, August 2005. IEEE.

[10] S. Han, B.-C. Lee, S. T. Yu, C.-S. Jeong, S. Lee, and D. Kim. Fold recognition by combining profile-profile alignment and support vector machine. *Bioinformatics*, 21(11):2667–2673, 2005.

[11] S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. In *Proc. Natl. Acad. Sci*, volume 89, pages 10915–10919, USA, 1992.

[12] E. Ie, J. Weston, W. S. Noble, and C. Leslie. Multi-class protein fold recognition using adaptive codes. In *Proceedings of the 22nd Intl Conf on Machine Learning*, Bonn, Germany, 2005.

[13] T. Jaakkola, M. Diekhans, and D. Haussler. A discriminative framework for detecting remote protein homologies. *Journal of Computational Biology*, 7(1):95–114, 2000.

[14] K. Karplus, C. Barrett, and R. Hughley. Hidden markov models for detecting remote protein homologies. *Bioinformatics*, pages 846–856, 1998.

[15] R. Kuang, E. Ie, K. Wang, K. Wang, M. Siddiqi, Y. Freund, and C. S. Leslie. Profile-based string kernels for remote homology detection and motif extraction. In *Proceedings of CSB 2004*, pages 152–160, 2004.

[16] C. Leslie, E. Eskin, and W. S. Noble. The spectrum kernel: A string kernel for SVM protein classification. In *Proceedings of the Pacific Symposium on Biocomputing (PSB)*, pages 564–575, 2002.

[17] C. Leslie, E. Eskin, W. S. Noble, and J. Weston. Mismatch string kernels for SVM protein classification. *Advances in Neural Information Processing Systems*, 20(4):467–476, 2003.

[18] S. Mallat. *A Wavelet Tour of Signal Processing*. Academic, New York, 2nd edition, 1999.

[19] K. Marsolo, S. Parthasarathy, and C. Ding. A multilevel approach to SCOP fold recognition. In *5th IEEE Symposium on Bioinformatics & Bioengineering (BIBE05)*, 2005.

[20] A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia. Scop: a structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol*, 247:536–540, 1995.

[21] S. Parthasarathy and C. C. Aggarwal. On the use of conceptual reconstruction for mining massively incomplete data sets. *IEEE Transactions on Knowledge and Data Engineering*, 2003.

[22] J. Platt. Fast training of support vector machines using sequential minimal optimization. *Advances in Kernel Methods - Support Vector Learning*, 1998.

[23] H. Rangwala and G. Karypis. Profile-based direct kernels for remote homology detection and fold recognition. *Bioinformatics*, 21(23):4239–47, 2005.

[24] H.-B. Shen and K.-C. Chou. Ensemble classifier for protein fold pattern recognition. *Bioinformatics*, Electronic Pre-Print, 2006.

[25] S. Y. M. Shi, P. N. Suganthan, and K. Deb. Multi-class protein fold recognition using multi-objective evolutionary algorithms. In *Proc. IEEE CIBCB*. IEEE, 2004.

[26] A. C. Tan, D. Gilbert, and Y. Deville. Multi-class protein fold classification using a new ensemble machine learning approach. *Genome Informatics*, 14:206–217, 2003.

[27] J. Weston, C. Leslie, D. Zhou, and W. S. Noble. Semi-supervised protein classification using cluster kernels. In *Advances in Neural Information Processing Systems (NIPS) 16*, pages 595–602, 2004.

[28] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2nd edition edition, 2005.