Information-based Biclustering for the Analysis of Multivariate Time Series Data

Kevin Casey[†]
[†]Courant Institute of Mathematical Sciences, New York University, NY 10003

August 6, 2007

Abstract

A wide variety of strategies have been proposed for the analysis of gene expression data, many of these approaches, especially those that focus on identifying sets of coexpressed (or potentially coregulated) genes, have been based on some variety of data clustering. While useful, many of these techniques have limitations that are related to the fact that gene expression data sets generally show correlations in both genes and conditions. Specifically, with respect to time series microarray data, often there are intervals of time in which gene expression is more more or less correlated for various sets of genes followed by intervals for which the correlation changes or disappears. This structure suggests that a clustering approach that partitions both genes and time points simultaneously might be profitable. Using techniques from information theory, one may characterize such a biclustering of the data in terms of how much various regions of the data set can be compressed. Motivated by these considerations, we have developed a biclustering algorithm, based on techniques from information theory and graph algorithms, that finds a partition of time course data such that sets of clustered genes are found for an optimally disjoint windowing of the dataset in time. This windowing places the boundaries between temporal regions at points in time for which gene expression activity undergoes substantial reorganization and thus sheds light on the process level dynamics in the biological data. We validate the method against more traditional techniques on both simple synthetic data as well as actual time course microarray data from the literature and show that while computationally expensive, our method outperforms others in terms of accuracy. Further, the method outlined here serves as a first step in the construction of automata based models of gene interaction as discussed in the paper [8].

Background

Clustering

Clustering data [16] is a major topic of research within the disciplines of statistics, computer science and machine learning. It is also an important practical technique used for data analysis and finds application in fields as diverse as biology, data mining, document understanding, object recognition and image analysis. Currently, many approaches under research are a response to biological questions related to microarray analysis of gene expression and genetic regulatory network reconstruction [25].

Clustering is characterized as a form of unsupervised learning wherein the elements of a dataset are organized into some number of groups. The number of groups might be known apriori or might be discovered during the clustering process. Clustering is generally regarded as a form of statistical learning executed without the benefit of known class labels (i.e. unsupervised), that finds a natural partition of a dataset that is optimal in some respect (where optimality makes sense in the context of the data under consideration). Such techniques contrast with classification achieved by supervised methods, in which a set of labeled data vectors are used to train a learning machine that is then used to classify novel unlabeled patterns. Clustering methods discover groups in the data without any advanced knowledge of the nature of the classes (except perhaps their number) and are thus entirely data driven. Less abstractly, clustering is the classification of feature vectors or patterns into groups when we do not have the benefit of patterns that are marked up with class information. It is generally exploratory in nature and often makes few assumptions about the data. When clustering, one is interested in finding a grouping (clustering) of the data for which elements within the same group are similar and elements associated with different groups are different (by some measure). There are a wide variety of methods and similarity measures that are used to this end, some of which are discussed below.

Algorithmically, clustering methods are divided into a number of classes (e.g. hierarchical, partitional, model-based, density-based, grid based, etc. [16]). More recently, a number of techniques based on spectral methods and graphs [27] have also been developed, as well as methods relying on information theoretic principles. Additionally, clustering may be hard (each pattern belongs to a single group) or soft (each data vector belongs to each cluster with some probability). Hierarchical clustering (common in biological applications) produces a nested family of more and more finely grained groups, while partitional clustering finds a grouping that optimizes (often locally) some objective function. As mentioned above there are a wide variety of clustering techniques that have been developed, our technique is an iterative partitional method that use techniques from information theory to formulate the optimization problem and that expects data in the form of a multivariate time series.

Additionally, there has been much recent interest in so called biclustering techniques (see below), especially in the context of biological research. This report focuses on methods related to partitional algorithms, and the current work can be seen as a relative of biclustering algorithms that partition data by optimizing some measure of energy or fitness. What is unique to this work is the emphasis on creating a partition specifically for ordered temporal data (i.e. time series), and characterizing the partition using the language of lossy data compression. In our specific case, we are looking for a windowing or "segmentation" of a time series dataset into intervals, within each of which we perform a clustering. In this way we achieve a biclustering of our data for which each window of the data is optimally clustered.

Partitioning algorithms divide (i.e. partition) the data into some number of clusters such that some measure of the distances between the items in the clusters is minimal while the dissimilarity between the clusters is maximal. The number of clusters is usually specified by the user, but there are techniques for automatically discovering model size (see below). Examples of partitioning algorithms include the popular

k-means and k-medians algorithms.

The present work develops an iterative biclustering method that builds on previous partitioning algorithms, optimization techniques, and traditional graph search in order to find a set of partitions in both genes and time. It minimizes an energy term developed using the tools of information theory and results in a set of clusterings for a corresponding set of disjoint temporal windows that cover the dataset and share only their endpoints.

Within the computational biology community, clustering has found popularity as a means to explore microarray gene expression data, aiding the researcher in the attempt to locate groups of coexpressed genes (with the hope that coexpression might imply - at least in some circumstances - coregulation). However, this objective is difficult to achieve as genes often show varying amounts of correlation with different sets of genes as regulatory programs execute in time. It is this shortcoming that motivated the development of biclustering within the context of computational biology, and again there has been much work in this area. Part of the problem with many biclustering techniques however, is that they are computationally complex and they do not take special characteristics of the data into account. Our algorithm is specifically designed to work with time series data and to locate points in the data at which significant process level reorganization occurs. Furthermore, our technique differentiates small tight clusters, from large loose clusters of less related data elements, an important quality when dealing with biological data.

Historically, two important steps in any clustering task are pattern representation (possibly including feature extraction) and the definition of a similarity measure. Indeed these two tasks are related since the definition of distance between data elements may be seen as implicit feature selection (e.g. Euclidean distance treats distance in any component as equally important). We try to make few assumptions here other than that the data is temporal in nature (i.e. a time series) and that correlation captures proximity between vectors in a manner that we are satisfied with. Specifically, in what follows we present a model-free procedure for time series segmentation that makes no assumptions about the underlying distributions that generate our data. While we do rely on correlation as a measure of similarity, it should be pointed out in advance that we are not wedded to it and that one could choose to use another basis for distortion calculations if one preferred.

Finally, it is often the case that clustering procedures require a number of necessary parameters that must be supplied by the user. We have based our times series segmentation procedure on a clustering subprocedure that does not need any such additional inputs. As we will see, our algorithm attempts to search for the best values of such tuning parameters in a natural way.

Precision vs. Complexity

Discussions related to learning from data often begin with descriptions of curve fitting as an example that illustrates the trade-off one must make between precision and the complexity of data representation. If one fits a curve of too high a degree (high complexity representation), one risks over-fitting and an inability to generalize, whereas if one uses too low a degree (low complexity representation), one risks a poor description of the data. In unsupervised learning a similar problem is often met. Determining the appropriate model size and type are difficult enough when the data is labeled and such considerations become only more significant when one is dealing without the benefit of such information. The method below sidesteps the issue of model type by making few suppositions about the data, but the question of model complexity remains. How many clusters should one use to describe the data? We explore our specific solution to these problems in the discussion related to model size below, but make a few remarks now to clarify the issues at hand.

For our purposes, model complexity generally corresponds to the cardinality of out clustering variable $|T|$. The more we compress the data (i.e. the smaller $|T|$ or, as we will see, the lower $I(T;X)$), the less

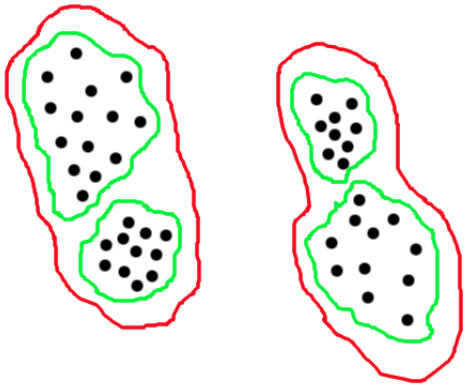

Figure 1: An example of the precision complexity trade-off. Which clustering better captures the basic structure of the data? The red clustering achieves more compression (i.e. has lower complexity) but losses its ability to discriminate the subclusters (i.e. has less precision). Conversely, the green clustering has greater precision at the cost of increased complexity. The decision about model size, here one between 2 or 4 clusters, will be discussed below.

precision and more expected distortion will obtain. Conversely, if $|T|$ is too large we might be modeling noise and overfitting which leads to a lack of ability to generalize to new examples.

The data analysis solution that we explore in this report allows for a consistent comparison of various descriptions of the data with respect to precision and complexity. In fact, the clustering subprocedure that we use, which is based on various recent extensions to rate distortion theory [10, 29], can be understood as an algorithm that captures just this trade-off, characterizing the problem in terms of similarity of clusters and compactness of representation (model size and amount of compression). In essence, one agrees to this trade-off and then attempts to do the best one can by formulating an objective function whose optimization finds the most compressed representation while striving for precision. This formalism offers a convenient way to define significant transition points in a large set of time series data as locations at which the amount of data compression one can do fluctuates. If one is motivated by a desire to capture biologically significant changes in such sets of data, this formulation is quite beneficial for it can be used to capture times for which biological processes undergo significant reorganization.

Microarray Data and Critical Time Points

Over the last decade, gene expression studies have emerged as a powerful tool for studying biological systems. With the emergence of whole genome sequences, microarrays (i.e. DNA chips) allow for the simultaneous measurement of a large number of genes, often the whole genome of an organism. When used repeatedly, microarray studies allow one to build up a matrix of expression measurements for which rows correspond to gene's expression levels and columns correspond to experimental conditions or time points for which a sample is taken. Thus, one may make comparisons between rows (vectors of expression

values for various genes), and columns (vectors of different gene's responses at a specific time or under a specific condition). In our study, we are interested in microarray data for which the columns are ordered in time, possibly (but not necessarily) at regular intervals. In this case, the rows are (possibly nonuniform) time series, that is, they are the expression profiles for the individual genes under study and they capture the history of the gene's dynamic behavior across time. A current area of research activity in computational biology is the effort to use such time course microarray data to elucidate networks of gene interaction, that is, to pull from the data a coherent picture of how groups of genes execute in a coordinated fashion across time and how the behavior of one group of genes influences and regulates the the behavior of other groups of genes. In our context we are interested in contributing to this general area of research by considering the points in time at which significant reorganization of gene expression activity takes place, for if we can locate these crucial points in time, we can aid biologists in focusing their analysis on the portions of the data that might be the most informative.

As indicated above, clustering has proved to be a powerful tool for data analysis and continues to be an active area of research. However when applied to microarray data, conventional clustering is somewhat limited. The problem derives from the fact that when analyzing a microarray data matrix, conventional clustering techniques allow one to cluster genes (rows) and thus compare expression profiles, or to cluster conditions (columns) and thus compare experimental samples but are not intended to allow one to accomplish both simultaneously. Often this becomes a problem, especially when one is attempting to track the development of groups of genes over time, that is, when the rows of the data matrix may be viewed as multivariate time series. In this case, biological intuition would suggest that as biochemical programs execute, various groups of genes would flow in and out of correlation with each other. That is, one would expect genes to show correlation with certain genes during some periods of time, and other genes during other periods. Additionally, there might be times when a gene's expression might not show a high degree of similarity with any other identifiable group of genes. For this reason, simply clustering genes across conditions (time points) does not make sense, as one would like to capture this dynamic aspect of the data. Moreover, one might even be explicitly interested in identifying times for which these critical points of gene expression reorganization take place. Locating such critical time points and understanding the gene activity related to them might shed light on network level arrangements and processes that are too difficult to discern when looking at all of the time points simultaneously.

Biclustering and Biological Data

Recently progress has been made on some of the limitations of applying clustering to microarray data analysis. Specifically, so called biclustering algorithms have been introduced that aim to find a clustering simultaneously in both the genes and columns of a data matrix. These techniques locate submatrices in the data for which subsets of genes exhibit correlated activity across subsets of conditions. There has been much research in this area in the recent past and several excellent reviews compile surveys of this work [19]. There are a substantial number of approaches to biclustering that result in various types of clustered data. Much of the work has centered on finding biclusters in microarray data for which the conditions are not necessarily ordered. We are interested in a specific type of clustering of our temporally ordered data, one that respects the ordering of the columns and that searches for blocks of time for which coordinated gene activity takes place. One assumption that we are working under is that the signals in our biological data show varying compression across points of critical reorganization. Here we are using compression in the technical sense found in the communication theory literature (as discussed below). While there has been some work concerned with finding biclusters in time series data (e.g. [34]), a biclustering algorithm that finds clusters of concerted gene activity within temporal windows that are optimal in the objective just men-

tioned (i.e. data compression) has not, to our knowledge, been investigated. Our interest is in a specific constrained biclustering problem for which the order of adjacent time points is respected. We have the twin objectives of clustering the data within temporal windows and deducing the correct window endpoints (i.e. "critical time points"). Thus, we offer a biclustering algorithm for time series microarray data that locates clustered gene activity in regions of time (i.e. windows) that are optimal in terms of the total amount of data compression that may be responsibly done on the data. Our biclustering produces clusters of genes in each of a number of disjoint temporal windows that partition the data in time. The windows are optimal in the amount of compression one can do on the underlying biological signals (i.e. expression profiles). We will clarify and make explicit these concepts in the discussion below.

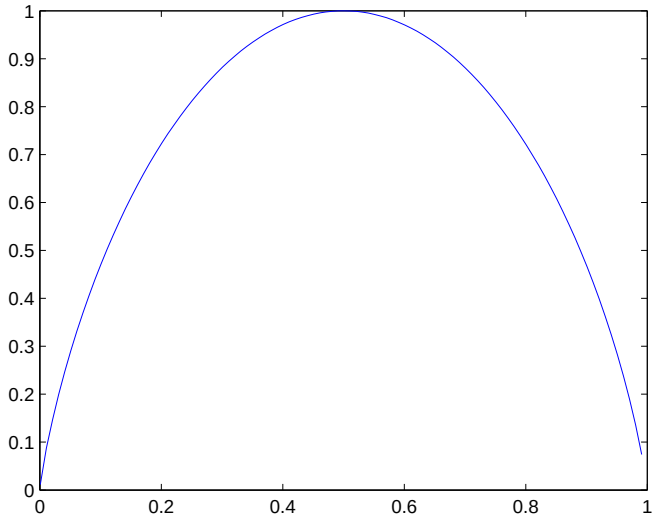

Figure 2: A simple example of entropy for the binary case. It is a plot of the binary entropy $H_2(x) = x \log \frac{1}{x} - (1-x) \log \frac{1}{(1-x)}$ as a function of x.

Theory

Information theory [10] is the standard tool for discussing data compression in a quantitative manner. Recently [32, 29], a novel approach to clustering that relies on information theory to characterize the precision vs. complexity trade-off has been introduced and applied to various data types (e.g. neural spike trains, microarray expression data, text). We will see that a method similar to these is suitable for use as a subprocedure for our time series segmentation algorithm as it offers one a measure of goodness that includes notions of both data compression and clustering fitness. First however, we must introduce some basic facts, definitions and concepts from the field of information theory.

Entropy

Information in the abstract is a difficult concept to define precisely. In the seminal 1948 paper [26] that both defined and solved many of the basic problems related to information theory, Claude Shannon defined the notion of entropy, which captures much of what is usually meant by "information":

$$H(X) \equiv -\sum_x p(x) \log p(x) = \sum_x p(x) \log \frac{1}{p(x)} \tag{1}$$

Entropy may be regarded as a measure of uncertainty (i.e. information) related to an event (or signal). It measures the expected number of bits (binary digits) required on average to describe the signal. In the context of data compression, the entropy is known to be the expected limit of lossless compression, that is, given a random variable X, one cannot generally compress X past $H(X)$ without loss. The definition above makes accessible a clear development of the concept of entropy based on the notion of "surprise". On an intuitive level, one would want a definition of information to capture what we experience subjectively as

surprise. That is, suppose one receives a message but one already knows the message's text. In this case there is no information transmitted via the message, as the receiver already knows what is being sent. Similarly, imagine observing a coin flip. If the coin is unbiased, with a 50 percent chance of coming up heads, then the coin flip is generally more surprising than a flip of a biased coin that has a 99 percent chance of coming up heads. There is more information conveyed by a flip of an unbiased coin than there is by a flip of the biased coin. (In fact, it can be shown that an unbiased coin maximizes the entropy (or information) related to the coin flip, and that the uniform distribution generally maximizes the entropy related to an event.) We can see then that we would like the information content of the coin toss to correspond to the amount of surprise one should expect upon the event taking place. But how does one characterize a surprising event? Generally, all other things being equal, one is more surprised when an event with a small probability takes place than when an event that is very likely happens. Thus, we want the information content of an event to be proportional to the inverse of the likelihood of the event $\frac{1}{p(x)}$, that way the less likely the event the more information is conveyed. Taking the expectation of the log of this quantity yields Shannon's concept of entropy $H(X)$, which is intended to measure the information content of a random variable X. Note that entropy as defined here is a functional, that is, a function of the distribution over X, rather than a function of a variable. One could also write $H[p(x)]$ to emphasize this point.

Additionally, one may define the entropy of two (or more) random variables, also known as the "joint entropy":

$$H(X,Y) \equiv -\sum_{x,y} p(x,y) \log p(x,y) = \sum_{x,y} p(x,y) \log \frac{1}{p(x,y)} \tag{2}$$

The joint entropy is the uncertainty (or information) associated with a set of random variables (in the above case two).

Finally, the conditional entropy is the expected uncertainty (or information) associated with one random variable Y, given that we know the value of another random variable X. That is:

$$H(T|X) \equiv \sum_{x} p(x) H(T|X=x) = -\sum_{x} p(x) \sum_{t} p(t|x) \log p(t|x) \tag{3}$$

These definitions are natural and follow the "chain rule", that is, the entropy of a pair of random variables is the entropy of one plus the conditional entropy of the other: $H(X,T) = H(X) + H(T|X)$.

KL Divergence and Mutual Information

The relative entropy or Kullback Leibler divergence (KL divergence) is a measure of the "distance" between two probability distributions, it is the expected logarithm of the likelihood ratio and is defined:

$$D[p||q] \equiv \sum_{x} p(x) \log \frac{p(x)}{q(x)} \tag{4}$$

One can immediately see that if the distribution $p = q$ for all x, then $D[p||q] = 0$. One can use the relative entropy to define yet another information measure called the mutual information. The mutual information is a measure of the amount of information that one random variable contains about another, and is defined:

$$I(X;Y) \equiv \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \tag{5}$$

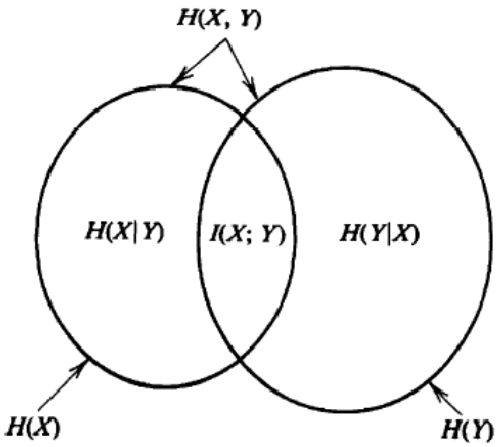

Figure 3: A pictorial representation of the various information quantities defined so far. Adapted from [10].

With the help of a little algebra it is not hard to show [10] that the following identities hold as well:

$$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) = H(X) + H(Y) - H(X,Y) \tag{6}$$

Thus, mutual information is the relative entropy between the joint distribution and the product of the marginal distributions. It is symmetric and always greater than or equal to zero [10]. It is equal to the uncertainty of one random variable left over after subtracting the conditional entropy with respect to another random variable.

In some sense, mutual information seems even closer than entropy to our colloquial notion of "information", since in most cases we speak of one thing containing information about another rather than just information in the abstract. This idea of shared information is exactly what mutual information formalizes and its role in what follows is crucial.

The following diagram, adapted from [10], is a useful pictorial representation of the the information measures that we have defined so far.

Rate Distortion Theory

In looking for the appropriate formalism to characterize our time series segmentation problem, it is useful to review rate distortion theory (RDT). Traditionally, RDT has been the main tool that information theorists use to address lossy compression in a rigorous manner. Given that clustering can be viewed as a form of lossy compression, and since the main component of our method is an information based clustering algorithm, it makes sense to review RDT and build on it as necessary. We will see that various recent extensions to RDT form the heart of our method and provide a powerful framework that we may use to attack our specific biclustering problem.

In rate distortion theory [10], one desires a compressed representation T of a random variable X that minimizes some measure of distortion between the elements $x \in X$ and their prototypes $t \in T$. Taking $I(T;X)$, the mutual information between T and X, to be a measure of the compactness or degree of compression of the new representation, and defining a distortion measure $d(x,t)$ that measures distance between "cluster prototypes" and data elements, traditionally in terms of Euclidean distance, one can frame this problem as a trade-off between compression and average distortion. The main idea is that one balances the desire to achieve a compressed description of the data with the precision of the clustering, as measured by the average distortion, and strikes the appropriate balance that maintains enough information while eliminating noise and inessential details.

In rate distortion theory, this trade-off is characterized mathematically with the rate distortion function $R(D)$, which is the minimal achievable rate under a given constraint on the expected distortion:

$$R(D) \equiv \min_{\{p(t|x):<d(x,t)>\leq D\}} I(T;X) \tag{7}$$

Where average distortion is defined to be:

$$<d(x,t)> = \sum_{x,t} p(x)p(t|x)d(x,t) \tag{8}$$

and is simply the weighted sum of the distortions between the data elements and their prototypes.

To find $R(D)$, we introduce a Lagrange parameter β, for the constraint on the distortion, and solve the variational problem:

$$\mathcal{F}_{min}[p(t|x)] = I(T;X) + \beta<d(x,t)>_{p(x)p(t|x)} \tag{9}$$

This functional captures the compression-precision trade-off and allows one to use an iterative method, based on Blahut-Arimoto [11, 4, 6] to calculate points on $R(D)$.

The solution to this problem [10]:

$$\frac{\partial \mathcal{F}}{\partial p(t|x)} = 0 \tag{10}$$

under the constraints $\sum_x p(t|x) = 1, \forall x \in X$ has the form:

$$p(t|x) = \frac{p(t)}{Z(x,\beta)} \exp^{-\beta d(x,t)} \tag{11}$$

where $Z(x,\beta)$ is a partition function, and the Lagrange multiplier β, is positive and determined by the upper bound on the distortion D:

$$\frac{\partial R}{\partial D} = -\beta \tag{12}$$

That is, the slope of the rate-distortion curve is $-\beta$. This is an implicit solution ($p(t)$ depends on $p(t|x)$) and is defined for a fixed set of prototypes. Different prototypes will change the solution obtained and for this reason selecting the correct prototypes is an important question. The joint optimization over cluster assignments $p(t|x)$ and prototypes is in general more difficult to solve and does not have a unique solution.

One can see from (11) that if the expected distance between a data element $x \in X$ and a prototype $t \in T$ is small, then the cluster assignment $p(t|x)$ will be large for that pair and x will be assigned to the cluster with centroid t. However, choosing these centroids so that one achieves optimal compression is a more complicated task and rate distortion theory unfortunately does not provide the solution.

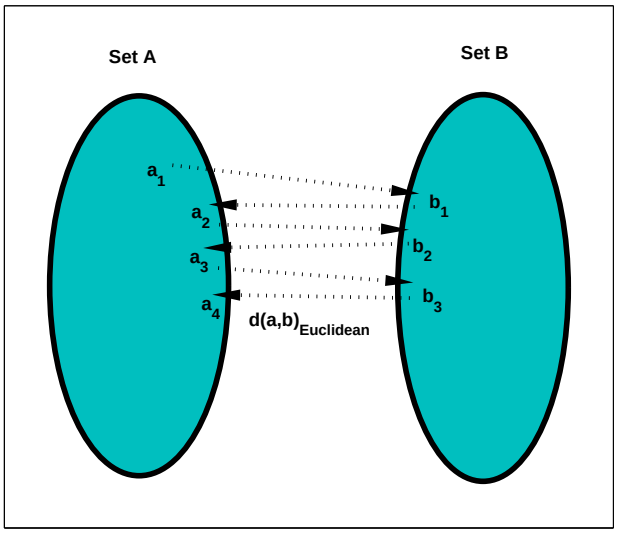

Figure 4: An example of the Blahut-Arimoto alternating minimization, in this case between two convex sets in $\mathcal{R}^2$ and the Euclidean distance. Since the minimization is of a convex function over convex sets, the algorithm is guaranteed to converge to the global minimum regardless of starting conditions.

We can calculate $R(D)$ using an iterative technique based on Blahut-Arimoto [11, 4, 6], in which we consider two convex sets and a convex distance function over them that is simultaneously convex in both of its arguments. We alternately minimize the distance between points chosen from the two sets, which has been shown to converge to the global minimum. An illustration of the procedure is provided in Fig. 4.

In the specific case of calculating the rate distortion functional, we define two sets:

1. A = the set of all joint distributions $p(t, x)$ with marginal $p(x)$ such that $<d(x, t)> \leq D$
2. B = the set of product distributions $p(t)p(x)$ with normalized $p(t)$

We can then reformulate the rate distortion functional $R(D)$ as the double minimization of the KL divergence between elements chosen from these sets:

$$R(D) = \min_{a \in A} \min_{b \in B} D_{KL}[a \parallel b] \tag{13}$$

We can rewrite $R(D)$ in this way because it can be shown that at the minimum, this KL divergence $D_{KL}[p(x)p(t|x) \parallel p(x)p(t)]$ equals $I(T; X)$, thus the D_{KL} bounds the information, with equality when

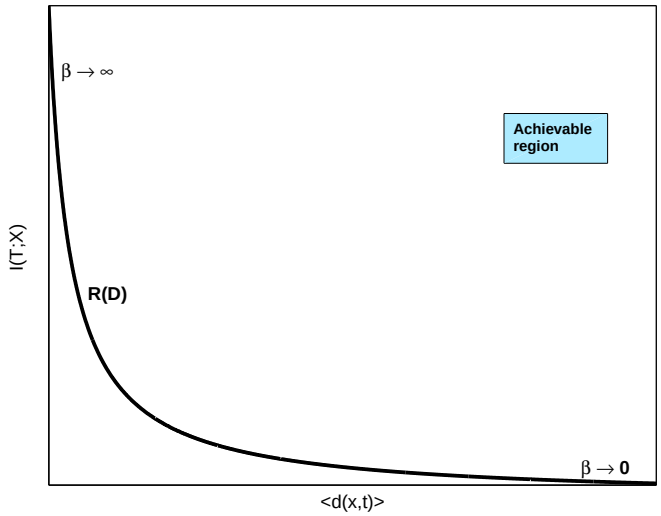

Figure 5: A typical rate-distortion curve, illustrating the trade-off between compression and average distortion. One can see that in order to achieve high compression (small $I(T;X)$) a larger upper bound on the expected distortion must be used.

$p(t)$ equals the marginal $\sum_x p(x)p(t|x)$. One can apply Blahut-Arimoto to sets A and B and $R(D)$ of (13). This allows one to fix β which, in turn, fixes the upper bound on the distortion D. We then pick a point in B and minimize the KL divergence $R(D)$, thus determining a point in $a \in A$. We subsequently minimize the KL divergence again, this time holding our point a fixed and generating a new point $b \in B$. We iterate this until the algorithm converges to a limit, which is guaranteed by [11]. Doing this procedure for various β values allows one to trace out an approximation to the rate distortion curve $R(D)$. An example of such a curve can be seen in Fig. 5. The points above the curve are the possible rate-distortion pairs, that is, they correspond to achievable amounts of compression for various upper bounds on the average distortion. We call this the "achievable region". The parameter β is related to the derivative of the rate-distortion function, and as one changes the value of β, one traces out the entire curve $R(D)$.

Information Based Clustering

From the discussion of rate distortion theory above, it is clear that one would like to have a formulation of the clustering problem that involves only relations between data elements, rather than prototypes. This would allow one to sidestep the thorny issue of how to correctly choose the cluster centers, which is one of the major drawbacks of conventional RDT. The information based clustering of [29] is just such a clustering scheme. Information based clustering is a method that is similar in many respects to RDT but that makes modifications to the distortion term that result in a number of important gains. The functional that characterizes information based clustering looks deceptively similar to RDT, but the distortion term masks an important difference. To perform information based clustering one minimizes the functional:

$$\mathcal{F}_{min} = I(T;X) + \beta<d_{info}> \tag{14}$$

This method replaces the $<d>$ term in the RDT functional with an overall measure of distortion $<d_{info}>$ that is defined only in terms of pairwise relations between data elements (rather than relations between data elements and prototypes). Here again β serves as a parameter controlling the trade-off between compression and precision, and sets the balance between the number of bits required to describe the data and the average distortion between the elements within the data partitions.

In information based clustering, $<d_{info}>$ is defined as the average distortion taken over all of the clusters:

$$<d_{info}> = \sum_{i=1}^{N_c} p(t_i) d(t_i) \tag{15}$$

Where N_c is the number of clusters (i.e. $|T|$) and $d(t)$ is the average (pairwise) distortion between elements chosen out of cluster t:

$$d(t) = \sum_{i=1}^{N} \sum_{j=1}^{N} p(x_i|t) p(x_j|t) d(x_i, x_j) \tag{16}$$

In the above, $d(x_1, x_2)$ is a measure of distortion between 2 elements in a cluster (this could instead be a measure for $m \geq 2$ elements, or some more complicated measure of multi-way distortion between elements). In our present case, we use a pairwise distortion measure (defined below) based on correlation.

The central idea is that one wants to choose the probabilistic cluster assignments $p(t|x)$ such that the average distortion $<d_{info}>$ is minimized, while simultaneously performing compression. This is accomplished by constraining the average distortion term $<d_{info}>$ and minimizing the mutual information between the clusters and the data $I(X;T)$ over all probability distributions $p(t|x)$ that satisfy the constraint on the compression level. The crucial difference between this method and RDT is located in the average distortion terms. For the example of pairwise clustering, we can easily see the difference. In RDT the average pairwise distortion is defined as:

$$<d_{RDT-pair}> = \sum_{i=1}^{N_c} p(t_i) \sum_{j=1}^{N} p(x_j|t_i) d(x_j, t_i) \tag{17}$$

Where the prototype t_i (the cluster centroid) is calculated by averaging over the elements in a single cluster:

$$t_i = \sum_{k=1}^{N} p(x_k|t_i) x_k \tag{18}$$

Whereas in information based clustering the average distortion is defined as:

$$<d_{Info-pair}> = \sum_{i=1}^{N_c} p(t_i) \sum_{j=1}^{N} \sum_{k=1}^{N} p(x_j|t_i) p(x_k|t_i) d(x_j, x_k) \tag{19}$$

The important thing to recognize is that in $<d_{RDT-pair}>$ the sum over k takes place before the call to $d(x_j, t_i)$ in the sense that the prototypes are calculated by averaging over members in the cluster as in equation (18). However, in $<d_{Info-pair}>$ the sum over k is outside of the call to $d(x_j, x_k)$. Thus, in RDT the distortion is pairwise between data elements and prototypes, whereas in information based clustering we have eliminated any reference to prototypes and only consider pairwise distortions between data elements.

For our purposes, the most important aspects of characterizing clustering in the above way are that there are explicit numerical measures of the "goodness" of the clustering (i.e. the average distortion $<d>$) as well as of the trade-off captured in the functional value. We can make use of these values to perform a

"segmentation" of our time series data such that we produce a series of time windows that capture transitions between major stages in the data or "interesting" events.

As in traditional rate distortion theory, in information based clustering one computes updates to the matrix of conditional probabilities $p(t|x)$ (i.e. the cluster assignments) by using an iterative procedure that calculates a Boltzmann distribution. Again, this method is based on Blahut-Arimoto and the form of the distribution is found by differentiating the clustering functional and setting it equal to zero. A proof for our case is provided below, however one should note that the form of the distribution contains an important difference that distinguishes it from traditional rate distortion theory. The form of the distribution is:

$$p(t|x) = \frac{p(t)}{Z(x,\beta)} \exp^{-\beta(d(t)+2d(x,t))} \tag{20}$$

This form is for a pairwise distortion measure and differs from (11) above in that it contains an additional term $d(t)$ (the average distortion for a cluster t), as well as a factor of 2 in the exponent. This form is a result of the differences in the original clustering functional and it adds an important notion of cluster "tightness" to the cluster assignment updating function. That is, tighter clusters (with low average distortion) are more desirable than diffuse clusters (high average distortion) and the clustering should try and produce clusters with low average pairwise distortion.

Time Series Segmentation

Given a set of time series gene expression data, we want to determine a sequence of windows in the dataset that capture important aspects of the temporal regulation of the sampled genes. We define a window, $W_{t_s}^{t_e}$ as a set of consecutive time points beginning at time point t_s and ending at time point t_e. Given a time series dataset with time points $T = \{t_1, t_2, \ldots, t_n\}$, the task is to segment the time series into a sequence of windows $\{W_{t_1}^{t_a}, W_{t_a}^{t_b}, \ldots, W_{t_k}^{t_n}\}$ such that each window represents some unique temporal aspect of the data. Note that adjacent windows meet at their boundary points but do not overlap. This problem is basically a special case of the biclustering problem discussed above, that is, we desire a biclustering that maintains the correct ordering of the elements in time but that finds clusters of data elements that are similar in informative temporal intervals. In the end, we have a number of windows, each with its own set of clusters. The clusters in each window are composed from the data subvectors that correspond to each window. The goal is to find the optimal such windowing that results in the maximal amount of data compression while preserving the important features in the data. The start and end points of such windows (i.e. t_s and t_e) correspond to points in the time series dataset where significant reorganization among genes has occurred. We would like to highlight such points in time, where the amount of compression changes significantly, for further investigation into the underlying biology.

We have attempted to create a method that relies on as few external parameters as possible while retaining flexibility. Thus, if one happens to have a good guess for the model size or temperature (i.e. β) parameters, then such values can be supplied. If no reasonable guess exists, we attempt to locate good values automatically (at additional cost in the running time). The one input that must be given, of course, is the distortion matrix that describes how similar various pairs of data elements are. We discuss the construction of this input below.

Distortion Measure Details

To create our distortion matrix, we used a pairwise similarity measure that is common in gene expression studies, the Pearson correlation coefficient [12]. While it has well known deficiencies (e.g. a lack of ability to capture nonlinear relations between the profiles that it is comparing), it also has various strengths, including its ability to function well as a measure of similarity between profiles that have a small number of points. Our approach is to form the distortion matrix directly from the values of the correlation coefficient:

$$d(i,j) = 1 - \frac{\sum_{n=1}^{N_p}(X_{i_n} - \bar{X}_i)(X_{j_n} - \bar{X}_j)}{S_{X_i} S_{X_j}} \tag{21}$$

Where $N_p = |X_i| = |X_j|$ and $S_X = \sqrt{\frac{1}{N_p}\sum_{n=1}^{N_p}(X_n - \bar{X})^2}$ is the standard deviation of X. We can calculate the (pairwise) distortion matrix based on the the correlation coefficients and feed this input into the clustering subprocedure of our time series segmentation algorithm. Here the values in the distortion matrix take 0 if the vectors are perfectly correlated and 2 if the vectors are perfectly negatively correlated.

If an objective measure of clustering goodness is required within the windows, one may measure the coherence [25] with respect to Gene Ontology terms, this gives us a good idea of how well the basic algorithm partitions the data with respect to an external qualitative grouping (we discuss this in further detail below).

Our notion of biological similarity is derived from the labels given in the Gene Ontology [5], and can be added to the distortion measure to augment distortion based purely on correlation of time series profiles. This allows us to validate our method in the manner common to gene expression clustering (i.e. by measuring coherence) and then to use these same ontology annotations to allow our algorithm to cluster based on both correlation of time series profiles as well as prior knowledge about the functional characteristics of the gene products themselves. Thus, we capitalize on the annotations provided by biological specialists as well as on the underlying characteristics of the data, and work toward automating a process that is ordinarily accomplished by hand (namely, choosing genes with specific known function and clustering around them to find potential functional partners).

Based on these concepts, we have begun to experiment with another related idea: namely, using an additional labeling (e.g. GO terms) in the clustering algorithm itself. Future work will include the construction of a similarity matrix that takes both correlation as well as proximity on the graph of GO terms into account. Initial experiments have included taking a weighted sum of distortion matrices, where one is a pairwise correlation matrix M_p with entries defined as in (21) above, and the other is a matrix M_g, with entries that correspond to how similar two genes' ontology labels are. Here both M_p and M_g have N rows (where N is the number of genes under consideration), and N columns. An entry e_{ij} in matrix M_g is in the interval $[0, 1]$ and takes on values closer to one the more the corresponding GO terms are shared between g_i and g_j in the ontology. The entry is zero if no terms are shared. When using this strategy, we create a distortion matrix by using a weighted combination of the above matrices: $M_s = aM_p + (1-a)M_g$ where $a \in [0, 1]$, and use M_s as the input to our clustering method. In fact, this method is quite general and can be used to add any type of prior similarity information we like to the algorithm. The difficulty here, of course, relates to choosing the relative weights on the various matrices appropriately and deciding how to weigh the contributions of the various ontology terms in M_g (i.e. more specific labels should count more than extremely general ones).

The flexibility of the distortion term also allows for prototypes to be selected by the user, thus forcing clusters to consolidate around specific profiles, this is useful if the researcher is interested in a single well-understood gene and wishes to find out what other genes might be related to it. In such a case, one would simply define a measure of pairwise distance that relied on how far apart genes' profiles were from some third target profile.

Previous work [29] has shown that using information measures (e.g. mutual information) to characterize distortion works well in practice. In our case we have steered away from this approach due to the short lengths of the time windows we would like to consider. With windows as short as four or five time points, estimating probability distributions well enough to calculate mutual information becomes too error prone.

Model Selection and Model Fitting

In addition to the question of how to generate the initial distortion matrix, there are also choices to be made about both the trade-off parameter beta and the underlying model complexity (i.e. the number of clusters N_c). Although these questions have been explored to some degree [23], including in the context of information bottleneck [30], we use a straight forward approach that favors simplicity and ease of interpretation in terms of rate-distortion curves. That is, we perform rudimentary model selection by iterating over the number of clusters while optimizing (line search) over beta. This procedure, while somewhat expensive, results in a fairly complete sampling of the rate-distortion curves (i.e. the plots of $I(X;T)$ vs. $<d>$) at various resolutions. Essentially, we trace the phase transitions (corresponding to different numbers of clusters) while tuning β and choose the simplest model that achieves minimal cost (and maximal compression) as measured by the the target functional. In this way, by optimizing the target functional over beta and the number of clusters, we obtain for each window a score that is the minimum cost in terms of model size and model fit, based on the trade-off between compression and precision. Obviously, for this method, run times can be substantial and for this reason we have developed an implementation that can take advantage of parallel hardware if it is available. We have used the Message Passing Interface [13], to provide parallel implementation on a cluster of machines. This offers the opportunity to decompose the larger problem into a set of clustering tasks to be performed on multiple machines and consolidated during the final stages of execution.

One aspect of the problem as we have formulated it above is worth mentioning here, that is, the relationship between β and the clustering solution produced by the clustering algorithm. We have stated that β parameterizes $R(D)$ and controls the trade-off between information preservation and compression. As β goes to 0, we focus on compression (in the limit we find just a single cluster with high distortion). Alternatively, as β goes to infinity, we focus on eliminating expected distortion (at the cost of increased mutual information). Thus, if we know before we run the algorithm, that we would prefer a very compressed representation of the data, we can set β accordingly. Similarly, if we know that we want to concentrate on minimizing distortion we can do that as well. We do not have to exhaustively search across β if we know what kind of solution we are looking for in advance, but if we want to try and determine the best possible minima, optimizing over this parameter is a reasonable task.

Graph Search for Optimal Windowing

Let $T = \{t_1, t_2, \ldots, t_n\}$ be the time points at which a given time series dataset is sampled, and l_{min} and l_{max} be the minimum and maximum window lengths respectively. For each time point $t_a \in T$, we define a candidate set of windows starting from t_a as $S_{t_a} = \{W_{t_a}^{t_b} | l_{min} < t_b - t_a < l_{max}\}$. Each of these windows may then be clustered and labeled with a score based on its length and the cost associated with the value of the clustering functional. Following scoring, we formulate the problem of finding the lowest cost windowing of our time series in terms of a graph search problem and use a shortest path algorithm to generate the final set of (non-overlapping) time windows that fully cover the original series.

To score the windows, we use a variant of the information based clustering procedure described above. We want to maximize compression (by minimizing the mutual information between the clusters and data

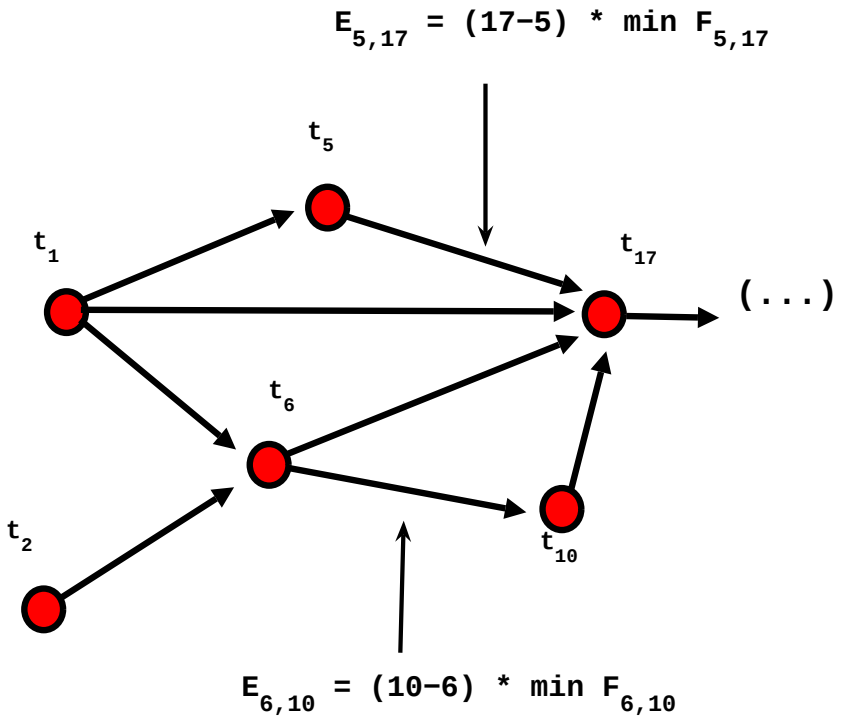

Figure 6: A portion of an example of the graph of weighted free energies, the output of the segmentation procedure. Edges are labeled with the clustering functional values weighted by window lengths. We use Dijkstra's algorithm to search for the minimum cost path through the graph (in the terms of the weighted free energy). In this way we find the lowest cost windowing of our data from the first time point to the last.

elements), while at the same time forcing our clusters to have minimal distortion. In such a framework, the measure of distortion is left up to the user, and while in the past the performance has been studied using a distortion term based on information estimation [28], we chose (due to the small lengths of our windows and the difficulty of accurately estimating mutual information between short sections of time series) to use a measure of similarity based on the Pearson correlation coefficient that is common in gene expression studies [12], and augment it with optional terms that measure similarity based on biological characteristics (as described above).

Once these scores are generated, we pose the problem of finding the lowest cost tiling of the time series by viewing it as a graph search problem. We consider a graph $G = (V, E)$ for which the vertices represent time points $V = \{t_1, t_2, \ldots, t_n\}$ and the edges represent windows with associated scores $E = \{W_{t_a}^{t_b}\}$ (see Fig. 6). The fully connected graph has N vertices $(t1, \ldots, tn)$ and n^2 edges, one between each pair of vertices. Each edge $e_{ab} \in E$ represents the corresponding window $W_{t_a}^{t_b}$ from time point t_a to time point t_b, and has an initially infinite (positive) cost. The edges are then labeled with the costs for the windows they represent, taken from the scores for the $\{S_{t_i}\}$ computed earlier, each edge cost gets $(F_{ab} \cdot length)$ where F_{ab} is the minimum cost found by the information based clustering procedure and length is the length of the window $(a - b)$. The edge weights are computed using a function that iterates over the number of clusters and optimizes over β and computes a numerical solution to equation (14) in an inner loop that tries multiple initializations and chooses the one that converges to the best cost. This algorithm is depicted in Fig. 7. In this way, we simultaneously label and "prune" the graph because edges that correspond to windows that have illegal length are left unlabeled and their costs remain infinite, while edges with finite cost are labeled appropriately. Our original problem of segmenting the time series into an optimal sequence of windows can now be formulated as finding the minimal cost path from the vertex t_1 to the vertex t_n. The vertices on the path with minimal cost represent the points at which our optimal windows begin and end. We may apply Dijkstra's shortest path algorithm to generate our set of optimal windows. We use the shortest path algorithm and generate a windowing that covers all of our original time points in a disjoint fashion and as such, segments our original time series data into a sequence of optimally selected windows which perform maximal compression in terms of the information based clustering cost functional. One thing to note is that if one desired to provide a set number of clusters or a specific beta based on some prior knowledge, one may easily do so. See Fig. 7 for a complete description of the segmentation algorithm in psuedocode.

Algorithmic Complexity

Dijkstra's algorithm is a graph search method with a worst case running time of $O(n^2)$ for a graph with n vertices. The clustering procedure used to score the windows is $O(N^3 \cdot N_c)$, where N is the number of time points in the window and N_c is the number of clusters. One can see this by noting that an outer loop of size N iterated over the rows of the conditional probability matrix and updates each entry (one for each of the N_c columns. Each update is of order N^2 since a call to the Boltzmann procedure, which generates the entries $p(t|x)$ in the matrix, must compute $d(t)$ the average distortion of cluster t, which contains two summations over N elements. This clustering procedure is nested in a loop that iterates over a small number of model sizes $O(1)[= constant \ll N]$ and a line search over potential values for β a $O(1)$ operation. This clustering procedure is run for each window of legal length, there are $\frac{n^2}{2}$ of these in the case of no restrictions on length. Creating the distortion matrix requires that we calculate the correlation coefficient for each of N^2 entries of the matrix where N is the number of genes in the dataset (larger than pairwise distortion measures would require many more computations). The graph search and distortion matrix creation complexity are dominated by the iterated clustering with a $O(N^5 \cdot N_c)$ cost.

Baseline Implementation

Motivated by a desire to provide a comparison between our information based approach and an existing clustering method that "discovers" the correct number of clusters, and so that we might have a fast segmentation algorithm for very large datasets, we have also implemented a baseline method for the time series segmentation that relies on the popular K-means algorithm [18] and the Bayesian Information Criterion (BIC) [24] for discovery of the correct number of clusters. Previous results [14, 21] indicate that a combination of K-means (or MoG) and BIC have worked well as a means to automatically discover clusterings along with the model size. However, such techniques do suffer from making assumptions about the process that generates the data. For example, in the case of K-means one is assuming data that can be fit by spherical Gaussians. That said, K-means is substantially faster than our iterative information based clustering subprocedure and for large datasets run times are substantially faster for the baseline implementation. For many simple test cases however, the information based methods recovered the correct number of clusters while the K-means BIC combination did not. Given the fact that the K-means implementation exhibits run times that are often many orders of magnitude faster, one approach to try in the future might be to seed the information based clustering with the solution obtained using K-means and BIC and then allow the information based clustering to converge to a more optimal solution if it exists.

Input:

Time series matrix with N series sampled over time points $[t_1, ..., t_n]$.
Range for the number of clusters, $N_c \in [L, ..., U]$, where $L, U \in \mathbb{Z}$.
Convergence parameter, ϵ

Output:

An optimal windowing of the data, each interval $[t_a, t_b]$ of which contains a soft partition of the N elements into $N_{c_{ab}}$ clusters.

Algorithm:

For each sub-interval $[t_a, t_b]$ of our time seres:
For every $\mathrm{x}_1, \mathrm{x}_2 = 1, ..., N$:
Create distortion matrix for this sub-interval, calculate $d(\mathrm{x}_1, \mathrm{x}_2)$
For every $\mathrm{N_c} = L, ..., U$:
Line Search Over β:
Initialize each row of the the conditional probability matrix with a random distribution.
$\mathcal{F}_{best} = \infty, m = 0$.
Loop:
For every $\mathrm{x} = 1, ..., N$:
For every $\mathrm{t} = 1, ..., N_c$:

$$p^{(m+1)}(t|\mathrm{x}) \leftarrow p^{(m)}(t) \exp\left\{-\beta[d^{(m)}(t) + 2d^{(m)}(t, \mathrm{x})]\right\};.$$

$$p^{(m+1)}(t|\mathrm{x}) \leftarrow \frac{p^{(m+1)}(t|\mathrm{x})}{\sum_{t'=1}^{N_c} p^{(m+1)}(t'|\mathrm{x})};.$$

$m \leftarrow m + 1$.
If $\forall\, \mathrm{x} = 1, ..., N,\ \forall\, t = 1, ..., N_c$ we have $|p^{(m+1)}(t|\mathrm{x}) - p^{(m)}(t|\mathrm{x})| \leq \epsilon$,
Break.

Evaluate $\mathcal{F}_{current} = I(T; X) + \beta < d >$, using the conditional probability matrix above.
If, $\mathcal{F}_{current} < \mathcal{F}_{best}$, then $\mathcal{F}_{best} = \mathcal{F}_{current}$.

Save $\mathcal{F}_{best_{ab}}$ for each sub-interval $[t_a, t_b]$.
Save $N_{c_{ab}}$, the model size for the best clustering on $[t_a, t_b]$.

Construct graph $G = (V, E)$, with vertices $V = \{t_1, t_2, \ldots, t_n\}$ and edge weights $E_{ab} = N_{c_{ab}} \cdot \exp^{\mathcal{F}_{best_{ab}}} \cdot (b - a)$.

Perform Dijkstra's algorithm over this graph to find the minimal cost windowing of the time series and the critical time points at the window boundaries.

Figure 7: Pseudo-code of the rate-distortion based algorithm. We iterate over the model size N_c, optimize over β, and repeat this procedure for different initializations, choosing the solution which minimizes the functional value. This clustering procedure is executed for each time window and the functional values are used to generate the cost graph.

Derivation of the Clustering Method

The derivation of the variational principle underlying our algorithm is similar to that of rate distortion theory [10], or the related information-based clustering [29]. There is a trade-off between the amount of compression one may achieve (i.e. the rate of the quantization) and the average distortion. The more bits we use to encode the representation, the smaller average distortion we can achieve.

This trade-off is captured through the modified rate distortion function, $R(D)$. The rate distortion function is defined to be the minimum rate (maximum compression) under a given constraint on the average distortion:

$$R(D) \equiv \min_{\{p(t|x):\langle d_{info}\rangle \leq D\}} I(X;T) \; . \tag{22}$$

That is, we want to minimize the number of bits used to encode the data, given a constraint on the distortion function (which for our purposes is a pairwise mapping: $d : X \times \tilde{X} \rightarrow \mathcal{R}^{+}$). The partitioning of X induced by the mapping $p(t|x)$ has an expected distortion, $<d_{info}>$, which is defined as the average pairwise distortion taken over all of the clusters:

$$<d_{info}> = \sum_{t} p(t)d(t) \tag{23}$$

Where the sum is taken over the number of clusters (i.e. $|T|$) and $d(t)$ is the average distortion between pairs of elements chosen from cluster t:

$$d(t) = \sum_{x_1}\sum_{x_2} p(x_1|t)p(x_2|t)d(x_1,x_2) \tag{24}$$

The expected distortion between x and a single member of cluster t is defined to be:

$$d(t,x) = \sum_{x_1} p(x_1|t)d(x_1,x) \tag{25}$$

The probability of a cluster t is:

$$p(t) = \sum_{x} p(t|x)p(x) \tag{26}$$

The pairwise distortion may be defined in any of a number of ways, depending on the data and the type of differences one wants to focus on. For example, we can take the pairwise distortion to be the Euclidean distance between two data vectors (as often done in vector quantization), or as the correlation (as popular in biological clustering applications), another possibility is to use the mutual information as in [29], which works well as long as one has a sufficient number of data points, but is not a good choice for our specific application.

To solve the rate distortion function we introduce a Lagrange multiplier β, for the constrained average distortion. Thus, we want to minimize the functional:

$$\mathcal{F}_{min} = I(T;X) + \beta<d_{info}> + \sum_{x}\nu(x)\sum_{t}p(t|x) \tag{27}$$

$$= \sum_{x}\sum_{t} p(t|x)p(x)\log\frac{p(t|x)}{p(t)} + \beta\sum_{t}p(t)\sum_{x_1}\sum_{x_2}p(x_1|t)p(x_2|t)d(x_1,x_2) + \sum_{x}\nu(x)\sum_{t}p(t|x) \tag{28}$$

over all normalized distributions $p(t|x)$. Where the last term in the expression corresponds to the constraint that $p(t|x)$ is a probability distribution. We carry this out with the understanding that:

$$p(t) = \sum_x p(t|x)p(x) \tag{29}$$

and thus that:

$$\frac{\delta p(t)}{\delta p(t|x)} = \frac{\delta \sum_x p(t|x)p(x)}{\delta p(t|x)} = p(x) \tag{30}$$

and that by Bayes' rule:

$$p(x|t) = \frac{p(t|x)p(x)}{p(t)} \tag{31}$$

To find the solution of the variational problem above we take the derivative with respect to our free variables (i.e. the $p(t|x)$'s). The solution,

$$\frac{\delta \mathcal{F}}{\delta p(t|x)} = 0, \tag{32}$$

for normalized distributions $p(t|x)$, takes the form:

$$p(t|x) = \frac{p(t)}{Z(x)} \exp\left[-\beta(2d(x,t) + d(t))\right] \ , \tag{33}$$

where Z is a normalization function:

$$Z(x) = \sum_t p(t) \exp\left[-\beta(2d(x,t) + d(t))\right] \ , \tag{34}$$

The Lagrange multiplier β, determined by the value of the expected distortion D, is positive and satisfies

$$\frac{\delta R}{\delta D} = -\beta \ . \tag{35}$$

Proof. Taking the derivative of (28) above with respect to the free variables $p(t|x)$ one obtains:

$$\frac{\delta \mathcal{L}}{\delta p(t|x)} = p(x) \log \frac{p(t|x)}{p(t)} + p(x)\beta[2d(t,x) + d(t)] + \nu(x) \tag{36}$$

To show (36), one breaks (28) up into three terms and takes the derivatives. Considering the three terms separately, we obtain:

$$\frac{\delta}{\delta p(t|x)} \left[\sum_x \sum_t p(t|x)p(x) \log \frac{p(t|x)}{p(t)} \right] \tag{37}$$

$$\frac{\delta}{\delta p(t|x)} \left[\beta \sum_t p(t) \sum_{x_1} \sum_{x_2} p(x_1|t)p(x_2|t)d(x_1,x_2) \right] \tag{38}$$

$$\frac{\delta}{\delta p(t|x)} \left[\sum_x \nu(x) \sum_t p(t|x) \right] \tag{39}$$

For (37) we have:

$$\frac{\delta}{\delta p(t|x)}\left[\sum_x \sum_t p(t|x)p(x)\log\frac{p(t|x)}{p(t)}\right] \tag{40}$$

$$= \sum_x \sum_t \left[\frac{\delta}{\delta p(t|x)} p(t|x)p(x)\log p(t|x) - p(t|x)p(t)\log p(t)\right] \tag{41}$$

$$= \sum_x \sum_t \left[\frac{\delta}{\delta p(t|x)} p(t|x)p(x)\log p(t|x)\right] - \sum_x \sum_t \left[\frac{\delta}{\delta p(t|x)} p(t|x)p(t)\log p(t)\right] \tag{42}$$

$$= p(x)[1+\log p(t|x)] - p(x)[1+\log p(t)] = p(x)\log\frac{p(t|x)}{p(t)} \tag{43}$$

That is, the first term of (36).
For (38) we have:

$$\frac{\delta}{\delta p(t|x)}\left[\beta\sum_t p(t)\sum_{x_1}\sum_{x_2} p(x_1|t)p(x_2|t)d(x_1,x_2)\right] \tag{44}$$

$$= \beta\sum_t \left[\frac{\delta}{\delta p(t|x)} p(t)\sum_{x_1}\sum_{x_2} p(x_1|t)p(x_2|t)d(x_1,x_2)\right] \tag{45}$$

Fixing t and using the product rule,

$$= \beta\left[\frac{\delta p(t)}{\delta p(t|x)}\sum_{x_1}\sum_{x_2} p(x_1|t)p(x_2|t)d(x_1,x_2) + p(t)\frac{\delta}{\delta p(t|x)}\sum_{x_1}\sum_{x_2} p(x_1|t)p(x_2|t)d(x_1,x_2)\right] \tag{46}$$

$$= \beta p(x)[2d(t,x) + d(t)] \tag{47}$$

That is, the second term of (36).
For (39) we have:

$$\frac{\delta}{\delta p(t|x)}\left[\sum_x \nu(x)\sum_t p(t|x)\right] \tag{48}$$

$$= \sum_x \sum_t \left[\frac{\delta}{\delta p(t|x)}\nu(x)p(t|x)\right] \tag{49}$$

Fixing x and t,

$$= \left[\frac{\delta}{\delta p(t|x)}\nu(x)p(t|x)\right] = \nu(x) \tag{50}$$

That is, the third term of (36).
Setting (36) equal to zero, letting $\log Z(x) = \frac{\nu(x)}{p(x)}$ and factoring out the $p(x)$ we have,

$$p(x)\left[\log\frac{p(t|x)}{p(t)} + \beta[2d(t,x)+d(t)] + \log Z(x)\right] = 0 \tag{51}$$

this implies,

$$\log \frac{p(t|x)}{p(t)} Z(x) = -\beta[2d(t,x) + d(t)] \tag{52}$$

taking exponentials and rearranging we obtain the solution (33):

$$p(t|x) = \frac{p(t)}{Z(x)} \exp\left[-\beta(2d(x,t) + d(t))\right] \ , \tag{53}$$

Chapter 13 of [10] illustrates the proof that $0 \leq \beta$, which follows from the convexity of the rate distortion function. The relationship between $R(D)$ and β from Eq. (35) is also discussed in the reference.

□

Evaluating the Clustered Windows

One important question that comes up when using the techniques described above to locate interesting time points is how one should best determine the quality of the segmentations produced by the algorithm. We save the complete discussion of this problem for following sections and focus here on characterizing the quality of the clusterings that are produced by the information based subprocedure. That is, all things being equal, how may one compare two different clusterings of the same subset of our time series data? Here, two considerations are helpful. The first approach is to visualize the trade-off between compression and distortion with the help of rate-distortion curves (commonly used in information theory). This visual depiction provides a global understanding of the output of the algorithm across a range of parameter values. The second approach is to complement such visual insights with measurements of the "objective" quality of the clustering by comparing it to "hand made" qualitative groupings of the data produced by experts. A method that has gained some popularity for this purpose in biological applications is the "coherence" [25].

Rate-distortion Curves

It is illustrative to plot the trade-off curves for various numbers of clusters to see how the clustering functional behaves over a range of conditions. In Fig. 8 we plot these curves for $|T| = 2, 3, 4, 5, 7, 10, 15, 20$, this is for a clustering of the restricted Tu dataset discussed below. In the lower right hand corner of the figure, one can see the green curve corresponding to $N_c = 2$ (that is, $|T| = 2$). Obviously, for such a limited precision, the average distortion will remain high (assuming the actual number of clusters in the data is greater than 2). The next curve to the left (dark blue with diamond glyphs), has far greater performance. This curve, representing three clusters, achieves a much lower average distortion as $I(T;X)$ increases. This improvements is due to the fact that the majority of the structure in half of our test dataset falls into 3 basic categories (genes corresponding to Ox, R/B and R/C). At the other extreme, if one is willing to use 20 clusters (cyan curve on far left), one can achieve significantly lower average distortion, in this case the compression is not as significant (i.e. higher $I(T;X)$). For the form of the trade-off functional we use, $-\beta$ corresponds to the slope of the compression-distortion curve. Note that the upper left corner of the figure corresponds to $\beta \to \infty$, whereas the lower right corner of the figure corresponds to $\beta \to 0$. As we increase β the clusters become more deterministic, with most of the probability mass occupying a specific $p(t|x)$ for each x. Conversely, as we decrease β the clustering becomes "softer", with each x being identified with a variety of clusters with nonzero probability. It is also important to note that as we increase β the curves begin to saturate, achieving a minimal distortion that depends less and less on the number of clusters used. In the limit, once $|T| > |X|$ adding additional clusters is no longer significant, as each x is already identified (for the most part) with its own cluster.

Rate-distortion curves offer a good way to visually summarize the trade-off between the goodness of the clustering and the amount of data compression, however they are completely quantitative in the sense that they only use the numerical characteristics of the data. In the next section we briefly discuss a technique that has been used to characterize the performance of a clustering method's ability to produce the "correct" qualitative description of the data (i.e. the correct clusters).

Coherence

A technique known as measuring the coherence [25] of a clustering is a good way to assess the overall quality of a clustering method. That is, one wishes to ensure that a clustering method produces the same qualitative groupings that a human expert would while analyzing the same data. Slonim et al. [29] have

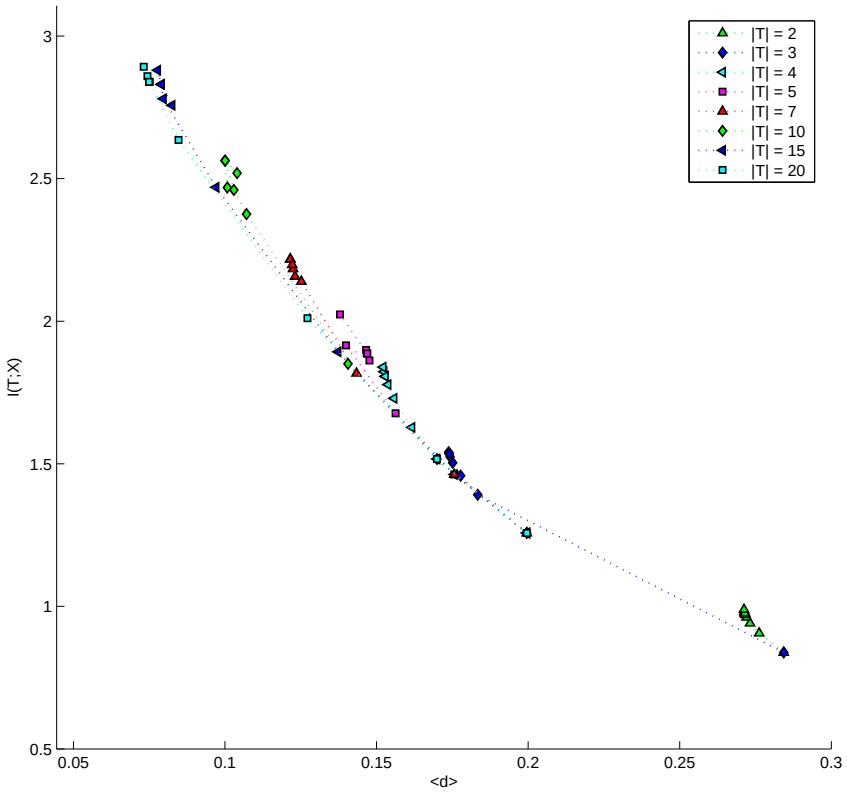

Figure 8: Compression vs. distortion curves for a subset of the Tu dataset for various numbers of clusters ($|T| = 2, 3, 4, 5, 7, 10, 15, 20$). The vertical axis represents $I(T;X)$ (i.e. compression), while $<d>$ (i.e. average distortion) is plotted on the horizontal axis. For a set number of clusters, a single curve traces out the trade-off between compression and information preservation, for example, the bottommost cyan curve represents the trade-off for 20 clusters. Note how as one increases the cardinality of T, the curves begin to saturate along the $<d>$ axis. That is, there is a lower bound on the size of the average distortion and even if one is willing to use a large number of clusters (high I(T;X)) ones minimal distortion is bounded.

used coherence to validate the general strategy of information based clustering and have shown that such clustering techniques produce good qualitative groupings when compared to those produced by a human (and indeed outperform many other popular clustering algorithms). The basic idea of evaluating coherence is that one looks at the ratio of "enriched" items in a cluster over the total number, where an item is enriched if it is labeled with an annotation that occurs with a significantly high frequency (as measured using the hypergeometric distribution, P-values and the Bonferroni correction [29, 25]). In our case, this technique is useful for determining the relationship between our time windows and biological events that they might correspond to. For example, we can use the MIPS FunCat database to determine whether our windowing of the Tu dataset (described below) maps cleanly onto the characteristics of the data described previously in the literature. Or we can use the GO biological process annotations to describe our temporal segmentation of the data in terms of biological events.

Results

We have tested our method on both simple synthetically generated time series, as well as real biological data, and have confirmed that our clustering procedure does indeed locate important temporal structure in the data. We mention that the present work is preliminary and that the overall motivation of the segmentation algorithm is related to the larger goals of developing automata like models from biological data [8] that one may use in further activities (e.g. model checking [9] and hypothesis generation/investigation).

Rather than simply focusing the biologist's attention on sets of genes, our central aim is to produce a process level summary of the aggregate gene expression. Our focus is on determining points in time during which processes undergo significant reorganization and on grouping coexpressed processes together between such points. This emphasis underscores the main assumption of this work: that at critical time points the amount of compression that can be effectively accomplished on the data is likely to fluctuate, and that such situations will be evident in the values taken by the optimum of the clustering cost functional across temporal windows. The clustering functional captures the goodness of the clustering (in the average similarity term) as well as the amount of compression (in the mutual information term). Its optimal value, taken over various model sizes and "temperature" parameters (i.e. β's), represents the best compression one can achieve while striving for an accurate (i.e. minimal distortion) representation of the underlying data. At locations where this optimal value changes, or where it corresponds to a different sized model, some biological reorganization could likely be taking place.

We are concerned with finding the time points at which such fluctuations occur and relating them back to biological facts and hypotheses. Further, we would eventually like to move past simple clustering to develop automata based models of gene expression, as well as tools that we can use to reason about and validate various hypotheses with respect to the data. We have reported some preliminary efforts in this direction in [8, 17], but the bulk of the formal methods have yet to be developed. We would like to add however, that the time series segmentation below may be seen as a data processing step on the way to building more complicated models of biological systems (e.g. in the context of the GOALIE project [3]). As a preliminary step in this direction, we use hybrid automata as generative models to create our synthetic test cases, and attempt to use our segmentation algorithm to extract the partition of the data that most closely matches the graphical structure of the automaton that generated the test case. We discuss the strategy in detail below but we should underscore that this approach allows us to validate our segmentation algorithm on cases for which a "correct" partition is known (as the structure of the hybrid model that generates the synthetic data defines the appropriate segmentation of the generated data set). To evaluate how closely the segmentation our information-based algorithm outputs matches the correct partition, we again turn to the tools of information theory, and characterize the distance between various alternatives in the space of segmentations using a metric based on mutual information. We discuss this metric and the space of segmentations below, and show that our distance measure is in fact a genuine metric.

We use synthetic data to validate that our algorithm works well and then demonstrate its usefulness on a recently investigated set of cyclic metabolomic data from yeast [33], as well as a life-cycle data set related to malaria [7]. In the case of synthetic data, we have focused our attention on verifying that simple examples of the types of events we would like to determine are indeed located by our algorithm. We generate these cases using hybrid automata augmented with splines and noise distributions. In the context of the biological data (taken from yeast and malaria gene expression microarray experiments previously reported in the literature), we extracted interesting temporal structure directly from the data without the use of any of the additional methods that are conventionally used, nor any other initial preprocessing of the time series. In doing so, we have replicated some results from earlier papers [7, 33] using techniques that are altogether different

from those employed in the original analysis. The two examples below both involve determining structure in the data using only our time series segmentation algorithm. The work agrees strongly with independently reported results in the literature found using frequency based analysis [7, 33] and has been written up for publication [31, 17].

Generation of Synthetic Data via Hybrid Automata

We would like to determine the sensitivity of our algorithm to noise present in the data. That is, we want to determine how the quality of the critical points and clusterings that our algorithm produces vary as the time series become more and more corrupted by noise. To investigate this, we require a means to generate datasets that are similar, yet are in some sense "noisy copies" of each other. A typical example of such data is illustrated in Fig. 11. We would like to generate many different data sets in this manner, so that we may validate the quality of the partitions that our method outputs. In order to do this we use a construction called a *hybrid automaton*, which is a dynamical system with both continuous and discrete components. We use these hybrid automata, which we define shortly, as generative models, and rely on them to generate the various noisy copies of our data.

A *hybrid system* is a system with both discrete and continuous components. A hybrid automaton [15] is a formal model of such a system. In a hybrid automaton we model the mixed discrete and continuous dynamics of some hybrid system using a graph for which the discrete state of the system is modeled by the nodes of the graph, and the discrete dynamics, that is the transitions between discrete states, are modeled by the edges of the graph. The continuous state of the hybrid automaton is modeled by points in real coordinate space of some dimension (i.e. $\Re^n$) and the continuous dynamics are modeled by *flow conditions*, for example, differential equations that specify how the continuous state of the hybrid system being modeled varies continuously in $\Re^n$ for the discrete state in question. Each node in the graph determines a flow condition (e.g. differential equation, spline or some other model of continuous dynamics) and each edge of the graph may cause a discrete change in the state of the hybrid automaton via a *jump condition*, that is, some condition that when satisfied, causes the automaton to switch between discrete states. In our case we use simple hybrid automata with straightforward chain-like graphical structure to generate our data. Each type of data vector in our data set is generated by a noisy trace of the underlying hybrid automata. That is, we augment the flow conditions (in our case splines) at the nodes with noise distributions (in our case Gaussians with some specified standard deviation) so that we may generate noisy traces of our automata. The flow conditions specify the mean of the Gaussian for a certain node at a specified point in time.

The study of hybrid automata is an active area of research [15] and much work has gone into characterizing their use for applications such as model checking [9] and, more recently, systems biology [1, 22, 2]. Our present requirements are quite modest, we use hybrid automata to generate data sets for which a there exists (at least one) natural partition that is obviously correct. In this way, we can validate the success of our algorithm's attempts to extract the graphical structure of the underlying processes that generated the data by measuring the distance between the correct partition and the partitions that our segmentation algorithm produces.

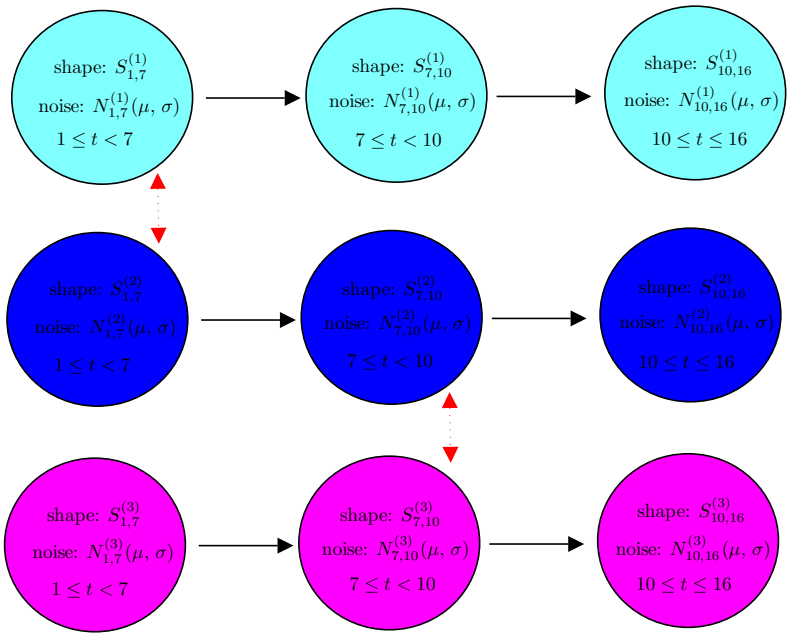

Figure 9: An example of three simple (noisy) hybrid automata that we use as generative models. The models themselves are standard, but we augment the automata with noise distributions at each node. The continuous dynamics at the nodes are generated via the "shapes" (i.e. splines or some other continuous functions that can be sampled at our time points of interest). The traces are made noisy by the addition of small values sampled from the noise distributions located at the nodes. We used the three models illustrated above to generate the data depicted in Fig. 11 by varying the variance of the noise distributions. In the figure above there are three hybrid models, each depicted in a color that corresponds to the color of the noisy traces in Fig. 11. The red arrows indicate nodes that share the same continuous dynamics modulo noise (i.e. the same underlying shape, for example $S^{(1)}_{1,7} \simeq S^{(2)}_{1,7}$).

Measuring Distances Between Segmentations via Mutual Information

Comparing two clusterings of the same underlying data is a difficult problem that may be approached from a number of perspectives. One may consider the assignment of pairs of data elements to various clusters in different clusterings (as in Rand's criterion [20] or the Jacard index [20]), such methods, which work by considering how pairs of data elements are grouped with respect to each other, have enjoyed great popularity as a means to evaluate how close or similar two different clusterings of the same data set are. Contrastingly, one may consider probabilities associated with the assignment of data elements to different clusters in the clustering solutions and use the formalism of information theory to quantify the relationship between various clusterings of the data. It is this second approach that we have followed, relying on the concepts of entropy and mutual information defined earlier to address the evaluation of proximity between various clusterings. In this way we can determine how well our method does with respect to the correct partition of the data. Of course, for real world data, there is not generally a "correct" clustering of the data, however we can do this in the case of our synthetic examples by construction. That is, because we use our augmented hybrid automata as generative models, the synthetic data sets have natural correct partitions that depend on the underlying automata that generated them. To reiterate, for our synthetic tests, there is a definite correct answer to the clustering problem, that is, our hybrid automata generate data sets for which a correct partition is known (since it is implicit in the construction of the data set via a hybrid automata of known structure). In what follows we will introduce an information theoretic technique from the literature, variotaion of information, that has been used sucessfully to compare individual clustering solutions. We will briefly state some known properties of this measure (e.g. that it is a metric on the space of clustering solutions). Finally, we will provide an extension to this technique which will allow us to measure distance between solutions to our time series segmentation problem, and show that our measure of distance, basically a sum of varioation of information terms, is a metric on the space of time series segementations (i.e. the space of cross products of clustering solutions).

In the discussion below we will need a few definitions, we consider them now. In what follows we consider a hard partition of the data for simplicity, but the discussion also applies, with minor adjustments, to soft partitions of the data as well.

A clustering T is a partition of a data set X into sets t_1, t_2, $\ldots t_k$. The t_i's are termed clusters. We can associate with each t_i a number n_i that is the number of datapoints that correspond to the cluster t_i in our clustering solution. Thus, if X contains n elements, we have:

$$n = \sum_{i=1}^{k} n_i \tag{54}$$

Additionally, we consider a second clustering T' of the same data, that is, a partition of a data set X into sets t'_1, t'_2, $\ldots$ t'_l. Many of the competing methods for determining distance or similarity between different clusterings of the same data rely on a contingency table (also referred to as a confusion matrix). This matrix is of dimension $k \times l$, where an element in the ith row and jth column is the number of points in the intersection of the clusters t_i and t'_j:

$$m_{ij} = |t_i \cap t'_j| \tag{55}$$

In order to use the tools of information theory to address the distance between various clusterings of a data set we need to discuss what our information measures, namely entropy and mutual information, mean with respect to clustering solutions. We begin by defining a measure of the amount of uncertainty in a cluster, that is, the entropy $H(t)$ of a cluster t. To do this, we need to define what probability distribution

we are working with. We want to capture the uncertainty related to the cluster assignment for a specific data element, that is, we want the probability that a data element x belongs to a cluster t_i. Assuming that each x has an equal chance of being picked, the probability of that data element being in cluster t_i is $p(t_i) = \frac{n_i}{n}$, that is, the ratio of $|t_i|$ to $|X|$, the total number of data elements in our data set. This yields a discrete random variable associated with the clustering T that takes on k values. The uncertainty or information associated with the clustering T is then:

$$H(T) \equiv -\sum_{i=1}^{k} p(t_i) \log p(t_i) \tag{56}$$

When there is only one cluster, the entropy is equal to 0, as the cardinality of the clustering increases, so does the uncertainty (assuming that each cluster has some of the data assigned to it). Additionally, the uncertainty increases as the data is distributed more uniformly over the clusters, rather than one cluster accounting for the majority of the probability mass. A clustering with data uniformly spread over two clusters has an entropy of one bit.

In order to define the mutual information we need some notion of joint probability between clustering solutions. In order to define this we use the elements in the previously mentioned contingency table. Following [20], we define the joint probability that a data element belongs to both t_i in T and t'_j in T' to be:

$$p(t_i, t'_j) = \frac{m_{ij}}{n} \tag{57}$$

This allows us to define the mutual information between the two clusterinfs T and T' as follows:

$$I(T; T') \equiv \sum_{i,j} p(t_i, t'_j) \log \frac{p(t_i, t'_j)}{p(t_i)p(t'_j)} \tag{58}$$

Where the index i runs over the elements in T and the index j runs over the elements in T'.

We know from our earlier review of information theory that the mutual information is non-negative and symmetric, further it is bounded by the minimum of the cluster entropies $H(T)$ and $H(T')$, with equality occuring when one clustering completely determines the other (i.e. when one clustering is obtained my merging some number of clusters from the other). Two clusterings are equal if and only if $I(T, T') = H(T) = H(T')$. In [20], Meila proposed the *variation of information* as a comparison criterion for two clusterings of the same data set.

$$VI(T; T') \equiv H(T) + H(T') - 2I(T; T') \tag{59}$$

Which by the definition of conditional entropy above implies:

$$VI(T; T') = [H(T) - I(T; T')] + [H(T') - I(T; T')] = H(T|T') + H(T'|T) \tag{60}$$

That is, $VI(T; T')$ measures the sum of the amount of information about T we lose and the amount of information about T' that we still need to gain, when moving from clustering T to clustering T'. In [20], various properties of variation of information are explored, including positivity, symmetry and the triangle inequality (i.e. VI is a metric on the space of clusterings of a given data set). We will see that one can readily extend these results to our problem of time series segmentation to achieve a distance measure between segmentations of a set of time series data, and that this natural distance (i.e. the sum of VI terms corresponding to individual time slices of the data set) is a metric on the space of possible segmentations.

One may consider the result of our time series segmentation problem (i.e. the clustered windows that are output by the search across the graph of weighted clustering functional values) as n clustered slices of the original data set. That is, assuming that the data is sampled at time points $1, 2, \ldots n$, then one may consider just the cluster assignments for slices of the data one time point in width. In order to measure the distance between two segmentations of the same data set (i.e. two different sets of clustered windows), we construct a confusion matrix for each slice of the segmentations and calculate the variation of information for each slice. We can then sum the per slice VI's across each segmentation and compare the values to each other. In general, let us assume that our data set is sampled at time points $t_1, t_2, \ldots t_n$ and that we have two segmentations (i.e. clustered windowings) S and S' of the data. We construct confusion matrices M^i and M'^i for each slice $[t_i, t_{i+1}]$ of the data and calculate the variation of information $VI(T;T')^i$ for each of the n slices using the definitions given above. The measure of distance between segmentations S and S', or the *segmentation variation* $SV(S,S')$, is the square root of the sum of the squares of the individual VI terms over all n time slices:

$$SV(S;S') \equiv \sqrt{\sum_{i=1}^{n} VI(T^{(i)};T'^{(i)})^2} \tag{61}$$

Where $T^{(i)}$ and $T'^{(i)}$ are the clusterings of the ith time slice found by segmentations S and S' respectively. This definition is motivated by the desire for a metric on the space of segmentations, which can be seen to be the product space of the clustered windows, or equivalently, the product space of the individual clustered time slices (since the clustered windows output by our algorithm are themselves simply cross products of contiguous time slices). Since a direct product of two metric spaces is a metric space (along with the square root of the sum of square distances in the original spaces, i.e. the metric in the new space), we define (61) in such a way that we obtain a metric space of time series segmentations, that is, the space in which S and S' live. In this way we immediately obtain from the definition of variational information:

$$SV(S;S') = \left(\sum_{i=1}^{n} [H(T^{(i)}) + H(T'^{(i)}) - 2I(T^{(i)};T'^{(i)})]^2 \right)^{\frac{1}{2}} \tag{62}$$

Or, rewriting once more in terms of conditional entropies:

$$SV(S;S') = \left(\sum_{i=1}^{n} [H(T^{(i)}) - I(T^{(i)};T'^{(i)}) + H(T'^{(i)}) - I(T^{(i)};T'^{(i)})]^2 \right)^{\frac{1}{2}} \tag{63}$$

$$= \left(\sum_{i=1}^{n} [H(T^{(i)}|T'^{(i)}) + H(T'^{(i)}|T^{(i)})]^2 \right)^{\frac{1}{2}} \tag{64}$$

The segmentation variation equals 0 if the segmentations S and S' are equal, otherwise it is positive. We can measure the competetiveness of two different segmentations S'_1 and S'_2 to a correct reference segmentation S by determining which of $SV(S'_1;S)$ and $SV(S'_2;S)$ is minimal. The smaller of the $SV(S'_i;S)$'s is the more correct segmentation of the given data.

Results that follow directly from the analysis in [20] include: $SV(S;S')$ is positive (this follows from the positivity of VI), symmetric (which follows from the symmetry of VI) and the triangle inequality holds (i.e. $SV(S_1;S_2) + SV(S_2;S_3) \geq SV(S_1;S_3)$. Thus, $SV(S;S')$ is a metric on the space of time series segmentations (i.e. on the cross product space of windows defined by $W_1 \times W_2 \times \ldots W_k$, where each

of the W_i's looks like $\Re^{n_i}$ for some integer n_i, where n_i is equal to the length of the ith window in the segmentation). Equivalently, since the W_i's are themselves cross product spaces composed direct products of time slices T_i, we could also say that SV is a metric on the product space of time slices. Further results we have include: the bounds $SV(S;S') \leq m \log n$ (for a data set with n elements sampled at m time points), as well as the property that the value of the segmentation variation depends only on the sizes of the clusters and not on the size of the data set. Many of the other results from [20] generalize to our segmentation problem as well as the extension to soft cluster assignments.

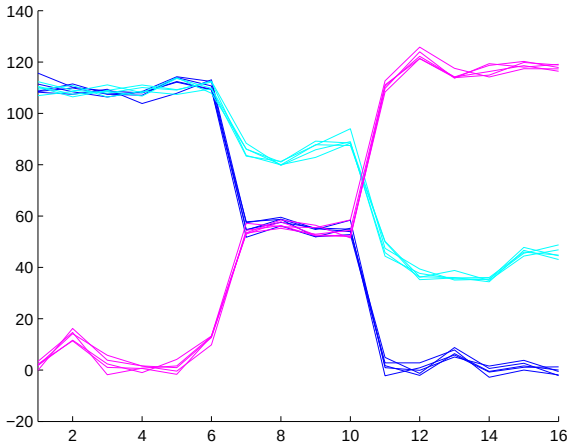

Figure 10: A simple test case composed of fifteen synthetic profiles used to test the information-based biclustering. Each color corresponds to a set of 5 randomly generated profiles, created by generating a trace from the underlying hybrid automata and adding Gaussian noise with a standard deviation of 2 to an underlying shape vector. Here one would expect a biclustering that finds interesting time points at t_1, t_7 and t_{10}, where the number of clusters in $[t_1, t_7]$ is 2, the number of clusters in $[t_7, t_{10}]$ is 2 and the number of clusters in $[t_{10}, t_{16}]$ is 3. This expected partition corresponds to the natural one obtained from the hybrid automata structure illustrated in Fig. 9.

Synthetic Data

In order to validate our method we constructed a number of test cases that illustrated many of the types of temporal events that we would like to extract from time series data. We discuss a representative example. As we are interested in situations in which new processes are initiated as well as situations in which some coordinated movement is occurring between groups of data elements, we created test cases that captured simplified versions of these types of scenarios. Ultimately, we would like to be able to extract networks of interaction and reason about them precisely, as such, a framework that focuses one's attention on the most informative regions in time is necessary. We begin by discussing our results on a simplified "toy" data set that, while trivial in structure, serves to illustrate the basic ideas behind the method and our testing strategy.

Consider the fifteen profiles in Fig. 10. They were created by generating three hybrid automata (Fig. 9)and sampling each five times while adding Gaussian noise to each component. Each color (blue, cyan, magenta) corresponds to five profiles, all generated from the same automata (and thus having the same underlying mean shape), but with different sampled noise subsequently added to the signal.

Further, the automata were constructed (i.e. the profiles have been generated) such that during some time intervals profiles show similarities that are not present during other intervals. For example during the interval $[t_1, t_7]$, the ten blue profiles have the exact same mean as the ten cyan profiles, although their noise components are different. Between t_7 and t_{10} the blue profiles fall to join the magenta vectors and between t_{10} and t_{16} they form a third cluster that is no longer similar to the other profiles. One characteristic that a quality biclustering of this data would show, is that at time points t_7 and t_{10} some interesting events take place that result in a change in the overall behavior of the profiles. Indeed, when we tested this example, our biclustering clustered the profiles as expected. In interval $[t_1, t_7]$ it found two clusters (the twenty blue and

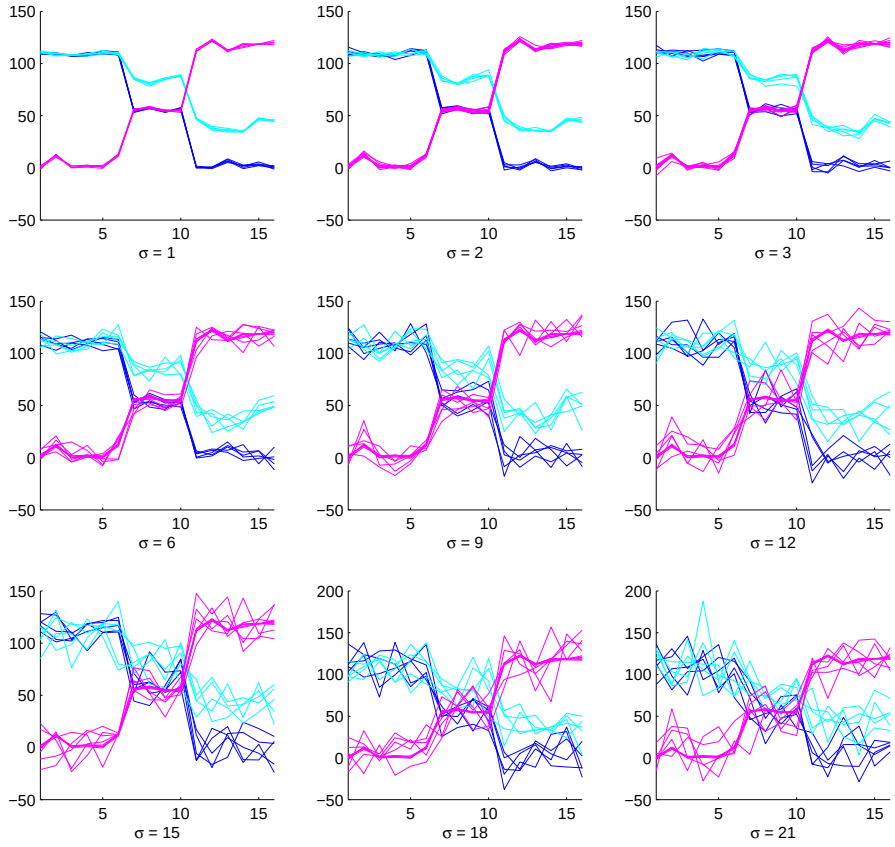

Figure 11: Additional examples of the previous 15 synthetic profiles generated by increasing the standard deviation of the noise distributions associated with the generative model. As before, each different color corresponds to a set of 5 randomly generated profiles, created by the addition of noise to the traces from the underlying hybrid automata used to generate the data. Again, one would expect a biclustering that finds interesting time points at t_1, t_7 and t_{10}, where the number of clusters in $[t_1, t_7]$ is 2, the number of clusters in $[t_7, t_{10}]$ is 2 and the number of clusters in $[t_{10}, t_{16}]$ is 3. However, as the examples become more noisy, we would expect the partitions to become more distant from the correct segmentation, especially as the clusters in the the interval $[t_7, t_{10}]$ become more difficult to discriminate between.

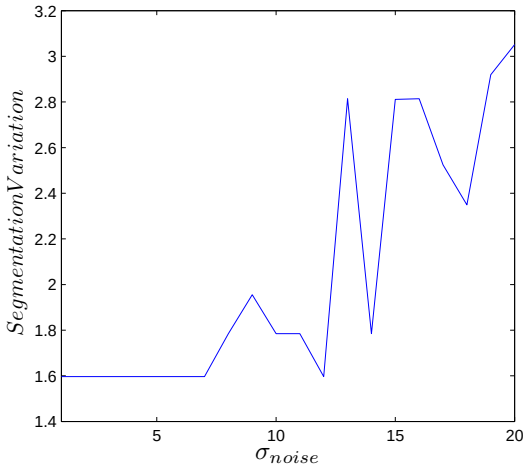

Figure 12: A plot of segmentation variation vs. variance of the noise distributions for the segmentations of the data sets illustrated in Fig. 11 (and additional data sets for values of σ not shown in the figure). As σ is increased we initially (i.e., for $\sigma \leq 7$) observe that the partitions found all have the same segmentation variation (i.e. the partitions converged to are the same up until a threshold of approximately $\sigma = 8$). As the data sets become progressively noisier, the segmentation algorithm has a more difficult time discerning the correct partition although there are a few places (e.g. at $\sigma = 12$ and $\sigma = 14$) where the partitions found are still fairly close to the correct one.

cyan profiles in one and the ten magenta profiles in the other), in interval $[t_7, t_{10}]$ it discovered two clusters and during time points $[t_{10}, t_{16}]$ it found three clusters as expected. Again, it is important to note that the number of clusters is not an input to the algorithm, but instead a parameter that is discovered (along with the cluster memberships $p(t|x)$) by the algorithm as it progresses.

In this case, each location in the underlying hybrid automata that generated the data had a Gaussian noise distribution with a standard deviation of 2 associated with it. This relatively mild noise, added component-wise to the traces from the model, allowed high quality partitions of the data to be generated with every run of the algorithm.

In order to study the robustness of our segmentation in the presence of noise, we consider various closely related data sets, each also with fifteen profiles, displayed in Fig. 11. These data sets were similarly created by generating three mean profiles (via the hybrid automata) and sampling each five times while adding noise. In these cases, however, we varied the variance of the noise distributions, creating versions of the original data set with increasingly corrupted signal.

Again, each color (blue, cyan, magenta) corresponds to five profiles, all generated using the same mean but with different sampled noise added to the signal. We would expect a good partition to divide the data as described above, however, it is unclear how the performance of the segmentation algorithm will suffer as we perturb the data with larger noise components. In Fig. 12, we plot the correctness of our segmentations (as measured by the segmentation variation) with respect to the standard deviation of the underlying noise distributions. We see that for larger values of σ performance does indeed suffer, however, the partitions obtained for modest amounts of noise are generally agree well with the correct solution.

We should point out that the above example is an extreme simplification of the type of data one might

meet in biological applications, especially given the large number of seemingly completely noisy vectors that arise in such data sets. We have, however, done experiments similar to the above but with large numbers of completely random noise vectors added to the data (e.g. greater than half of the data set). Generally, we obtain similarly accurate partitions of the data in such cases. We have chosen the above example for simplicity of discussion and graphical presentation, we will see the segmentation performance on larger, more difficult biological data sets below.

Although this synthetic example is trivial, it does represent a simplified version of the type of problem that we would like to address. That is, at some point an event takes place that changes the amount of compression that we can do responsibly on the data. We want to locate these points in time and generate a concise description of what is happening between them. It is worth mentioning that in this toy example, transitions between regions of differing amounts of compression often occurred in such a way that the model size actually changed as well. Such examples were chosen because they illustrate the key ideas most simply. As we will see when we apply the method to actual biological data, differences in the amount of compression that can be done often occur in situations for which the model size remains constant.

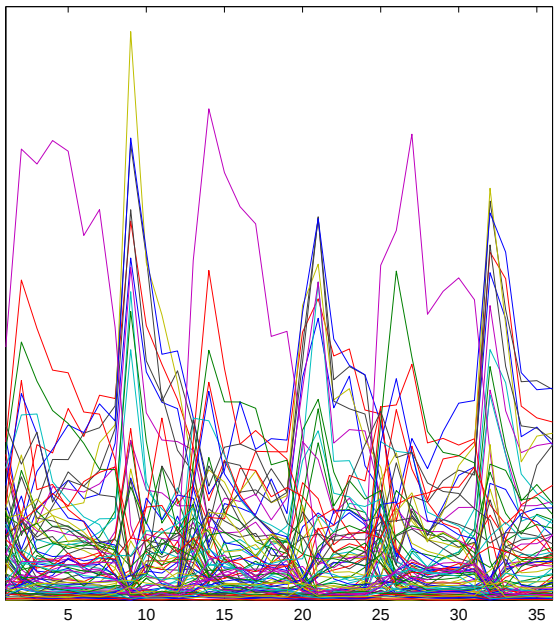

Figure 13: Plots of the most cyclic vectors (taken from [33]) with an equal number of additional noisy data vectors added.

Yeast Metabolic Cycle Data

Periodic behavior and biological clocks are common in living systems. Examples include the well known circadian rhythm, found across kingdoms, which facilitates coordination of organisms' wake/sleep rhythms with day/night cycles. The budding yeast Saccharomyces cerevisiae has been shown to exhibit other similar ultradian cycles in the form of glycolytic and respiratory oscillations. Recently, Tu et al. [33] used a continuous culture system to investigate a robust, metabolic cycle in budding yeast. They described a yeast metabolic cycle (YMC) that drives genome-wide transcription and coordination of cellular and metabolic processes in a manner they characterize as reminiscent of the circadian cycle.

We have applied our time series segmentation method to characterize the yeast metabolic cycle using the data of Tu et al. [33]. This data is a continuous growth-starvation-nutrition culture involving three cycles (sampled at 36 time points). The data capture the cyclic yeast respiratory activity, where cycles (as measured by oxygen consumption), were 4-5 hours in length (about 12 consecutive time points). Each cycle had a reductive, nonrespiratory phase followed by a oxidative, respiratory phase. The microarray data captures expression at intervals of 25 minutes over the three consecutive cycles. From this, Tu et al. determined the cyclic nature of the genes using a periodicity algorithm and determined a large number of genes that exhibited periodic expression. In the paper [33], autocorrelation based methods were used to capture the periodic nature of the data. The most common period was 300 minutes, however different genes were maximally expressed at different times. Genes encoding proteins related to energy, metabolism and protein synthesis were overrepresented in this group of periodic genes. Finally, cluster analysis was performed

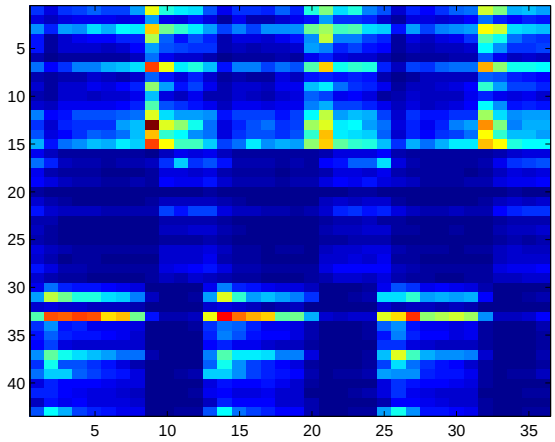

Figure 14: Heatmap of the most cyclic vectors (as reported in [33]).

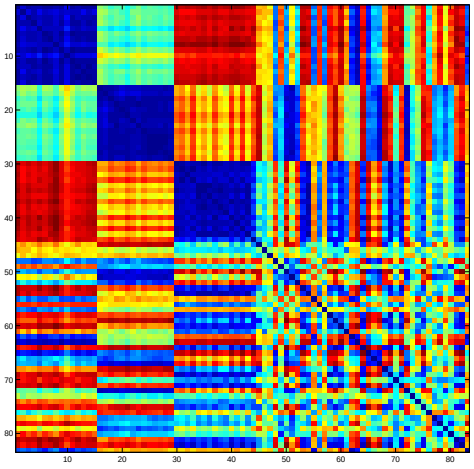

Figure 15: Difference matrix for a small test dataset composed of the most significant cyclic vectors from the Tu dataset along with an equal number of additional noisy data vectors. The cyclic vectors are in the top half of the matrix, one can clearly see this in the block structure.

in two ways; the first used the most periodic genes as cluster prototypes, an unbiased k-means was also performed.

One key result of the Tu et al. paper is the determination of three large clusters of expression patterns that oscillate in a coordinated manner throughout the phases of the metabolic cycle. The first cluster, called Ox (oxidative) takes place in intervals when dissolved oxygen has decreased and contains genes that code for amino acid synthesis, ribosome, sulfur metabolism and RNA metabolism. The second, called R/B (reductive, building) peaks when cells begin to increase oxygen consumption and involves genes for mitochondria, DNA replication, histones and spindle poles. Finally, the third cluster, called R/C (reductive, charging) is reported to encode genes involved in nonrespiratory modes of metabolism and protein degradation.

We would like to determine both these clusters and the critical time points in their genes' expression profiles directly from the microarray data, without having to specify interesting genes or perform preliminary analysis. Visually, we would expect critical points to correspond to times at which a cluster's coordinated expression changes dramatically.

Our biclustering proceeds by sequentially segmenting such time course data which is followed by determining the enriched functions in the windows corresponding to informative time intervals. With respect to the Tu dataset, we would like to use this information to characterize the relationships underlying the cyclic metabolic events. We attach biological significance to the clusters in each window by labeling them with their functional enrichments. From this, one can see that important time points corresponding to the periodicity of the groups of coexpressed genes are produced directly by our clustering algorithm rather than being produced via separate frequency analysis or by clustering around preproscribed gene expression profiles. Further, such analysis puts us in a position from which we can begin to think about building richer (e.g. automata based) models of the underlying biology.

Tu et al. report that expression for the Ox cluster peaked during time intervals [8-12], [20-24] and [32-36], expression for the R/B cluster peaked during intervals [10-14] and [22-26], whereas expression for the R/C cluster peaked during intervals [2-7], [14-19] and [26-31]. Our biclustering algorithm produced the segmentation [1-6][6-9][9-14][14-17][17-20][20-23][23-26][26-31][31-36]. These nine mined intervals correspond to the temporal windows for which expression peaked for each of the three clusters in each of the three metabolic cycles. Following time series segmentation, we label the mined windows with their functional enrichment (for the genes whose clusters' expression peaked in that window). The enrichment labels correspond to the categories described by Tu.

Again, it is significant to note that Tu at al. performed frequency analysis in addition to other data processing (clustering etc.) in order to pull out the biologically significant structure in the data. Our algorithm mines much of this structure in an automatic manner and, while there are still plenty of improvements to be made, we feel that such a completely automated approach, once perfected, will be a valuable tool in the pursuit of the construction of more complicated models directly from data. For example, in cyclic data like that of Tu, one might hope for an automata that captures the periodic nature of the data in terms of states and transitions between them. A biclustering, such as that produced by our algorithm, puts one in a position to build such models [8]. Additionally, such an approach could prove especially useful if the periodicity in the data is not as obvious as in the above yeast set.

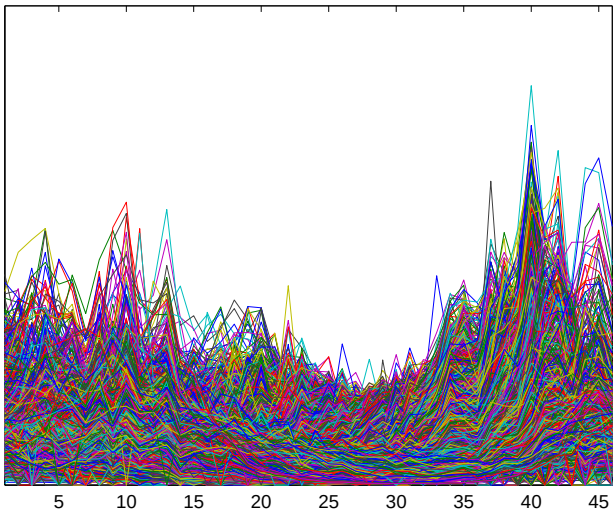

Figure 16: Plot of the Bozdech dataset (from [7]).

Intraerythrocytic Developmental Cycle of *Plasmodium Falciparum*

There are currently up to half a billion new cases of malaria reported annually. The parasite Plasmodium falciparum, one species of malaria-causing Plasmodium, is especially severe, resulting in as many as two million deaths each year and is responsible for the majority of the hundreds of millions of malaria episodes worldwide. While great gains have been made in the fight against malaria via drugs, vector control and advances in knowledge and public health, no solution to the disease has yet been found. With no present malaria vaccine, the disease continues to affect the lives and economies of many nations, taking a particularly devastating toll in many developing countries. The genome of P. falciparum, recently sequenced, will provide insight into the function and regulation of P. falciparum's over 5,400 genes and should bolster the search for future treatments and a possible vaccine [**?**].

Transmitted by mosquitoes, the protozoan Plasmodium falciparum exhibits a complex life cycle involving a mosquito vector and a human host. Once the infection is initiated via sporozoites injected with the saliva of a feeding mosquito, P. falciparum's major life cycle phases commence. These phases are: liver stage, blood stage, sexual stage, and sporogony. The blood stage is characterized by a number of distinct and carefully programmed substages which include the ring, trophozoite and schizont, which are referred to collectively as the intraerythrocytic developmental cycle (IDC).

We have used our information based time series segmentation technique in conjunction with other tools to investigate the dynamics of the intraerythrocytic developmental cycle of *Plasmodium Falciparum* [17]. In [7], Bozdech et al. study *P. Falciparum*, a recently sequenced strain of the human malaria parasite. The authors describe *P. falciparum*'s approximately 5,400 genes, the majority of whose functions are still unknown. It is understood that a large percentage of *P. falciparum*'s genome is active during the IDC and that the regulation pattern is such that as one set of genes is deactivated, another is being turned on, resulting in what the authors refer to as a continuous "cascade" of activity, whereby transcriptional regulation is controlled in a tightly synchronized manner. Using our clustering scheme we can reconstruct the main

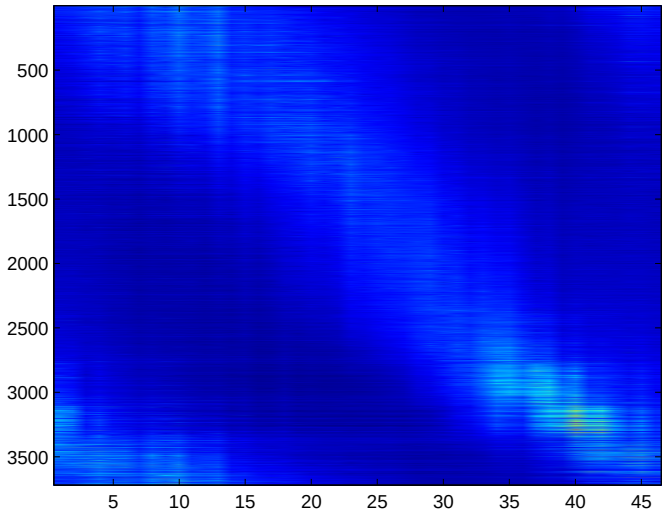

Figure 17: Heatmap of the Bozdech dataset, showing 3719 oligonucleotides over 48 hours (46 time points).

features of the system, including the cascade of genes, as well as the stages of the IDC and their associated processes.

Bozdech et al. conducted their investigation with the help of Fourier analysis, using the frequency and phase of the gene profiles to filter and categorize the expression data. They used the fast Fourier transform (FFT) to eliminate noisy genes and those that lacked differential expression. Most of the profiles registered a single low-frequency peak in the power spectrum, which the authors used to classify the expression profiles. Classified in this way, the cascading behavior of the genes involved in the IDC was clear. Our method reproduced this cascade of expression in an automated manner and without relying on frequency based methods, we were able to recover the underlying structure of the system using an approach based on our time series segmentation method. We segmented the time series in the "overview dataset" provided by Bozdech et al. [7]. This dataset contained 3719 oligonucleotides (represented by 2714 unique open reading frames (ORFs)). Bozdech provided an overview of the IDC transcriptome, by selecting all 3,719 microarray elements whose profiles exhibited greater than 70 % of the power in the maximum frequency window and that were also in the top 75 % of the maximum frequency magnitudes. Rather then using hierarchical clustering for analyzing the expression data, they addressed temporal order directly within the dataset. To accomplish this, the authors used the FFT phase to order the expression profiles to create a phaseogram of the IDC transcriptome of P. falciparum, (Fig. 17).

The windowing of the data, discovered using our information based segmentation method, corresponds well to the main stages of the *P. Falciparum* IDC as described in [7]. When the rate distortion clustering is run on the overview dataset, critical time points 7, 16, 28 and 43 drop out of the method as points at which the amount of compression that can be accomplished on the data significantly changes. These critical points signal times at which major functional reorganization of gene expression is likely to be taking place. Bozdech et al. note that the 17th and 29th hour time points correspond to the ring-to-trophozoite and trophozoite-to-schizont stages of the IDC, which agrees well with the results of our automated method. As one may verify visually from the plotted data (Fig. 16), notches in the aggregate profile of the expression

Window	Time period(in hours)	Number of Clusters	Stage
1	1-7	4	End of Merozoite Invasion and Early Ring
2	7-16	5	Late Ring stage and Early Trophozoite
3	16-28	4	Trophozoite
4	28-43	5	Late Trophozoite and Schizont
5	43-48	5	Late Schizont and Merozoite

Table 1: Clustered windows of the IDC data and their relationship to documented biological stages.

data occur at roughly these locations, which are also the locations found via frequency analysis [7] to be transitions between major functional stages (i.e. ring/trophozoite and trophozoite/schizont). The first critical time point produced by our clustering, at hour 7, corresponds to the end of the previous merozoite invasion. The last critical time point produced by our clustering, at hour 43, corresponds to the final portion of the schizont stage overlapping with the early portion of the next period. We again note that the analysis of Bozdech was carried out on a phaseogram (like that of Fig. 17), in which the data had been sorted based on the location of peak expression. Contrastingly, our method segmented the the data with no explicit notion of phase or frequency and extracted the same general structure.

Following the convention used in [17], the notation $W : C$ is used to denote the Cth cluster in the Wth window. We present the results of our segmentation in tabular form below (Table 2).

Window:Cluster	Description
1:1	This cluster is entering the ring stage. It is comprised of 631 ORFs and is labeled by ontology terms related to biosynthesis, glycolysis, and transcription.
1:2	This cluster is entering the ring stage. In this cluster there are 835 ORFs, which are primarily involved in translation and tRNA and rRNA processing.
1:0 and 1:3	Correspond to the end of the previous cycle
2:3 and 2:1	These clusters followed from 1:1 and 1:2, and correspond to the ring stage.
2:0	This cluster exhibits overlap from one stage to the next, illustrating the "cascade" of genetic activity, and is identified with the Early Trophozoite stage. This transition is comprised of 957 ORFs, which is in agreement to the 950 found by Bozdech et al.
3:3	This cluster contains 1400 genes involved in the ring stage, which is tapering off.
3:0	Contains Trophozoite ORFs (379)
3:2	Contains 1400 genes expressed later in Trophozoite stage
4:3 and 4:0	These clusters contain ORFs which were involved in the late Trophozoite stage
4:2	Contains ORFs expressed in the late trophozoite stage
4:1	Contains 669 ORFs that correspond to the beginning the schizont stage. 4:1 and 4:2 have a total of 1161 ORFs (compared to 1,050 as found by Bozdech et al.)
5:3	Comprised solely of ORFs from 4:2 and 4:1 (which at this point completing the schizont stage)
5:1	Contains 524 ORFs that are highly expressed in the late schizont stage and which have early-ring stage annotations. This is consistent with prior findings of "approximately 550 such genes" [7].

Table 2: Clusters from the IDC data and their biological descriptions.

Conclusion and Further Work

We have suggested an approach to automatically finding "interesting" events in a set of time series microarray data. Our technique relies on recently developed extensions to rate distortion theory, and more generally, on the formalism of information theory to characterize and describe such events. The key assumption that we make is that interesting time points are locations in the data where fluctuations occur in the amount of data compression that can be done. By this measure, we can find an optimal set of temporal windows across the data that capture the major changes in gene expression. Further, using some form of external labeling (e.g. GO or MIPS) we can tie these windows to biological events. We have demonstrated that our method shows promise, via performance on simple test cases and preliminary work with real biological data sets. Further, we have reproduced some results from the literature using techniques that differ considerably from those originally used.

Further work will include continued testing with more biological data to determine the strengths and weaknesses of the general method, as well as developing extensions to the approach with an eye toward building more complicated models of gene expression data. Moreover, we plan on developing an automated method that integrates ontology labels from various databases and makes them available to use in the clustering subprocedure as discussed above. Finally, we would also like to formalize and make more quantitative the nature of the agreement and disagreement between the results of our method and the methods of data analysis used in the original analysis of the yeast and malaria datasets described above.

I would like to thank Professor Bud Mishra for advising this project and offering support throughout the year. I would also like to thank Dr. Paolo Barbano for his time and consideration. Finally, I would like to thank my parents, whose support has been invaluable.

References

[1] Rajeev Alur, Calin Belta, Franjo Ivančić, Vijay Kumar, Max Mintz, George J. Pappas, Harvey Rubin, and Jonathan Schug. Hybrid modeling and simulation of biomolecular networks. *Lecture Notes in Computer Science*, 2034:19–??, 2001.

[2] M. Antoniatti, B. Mishra, C. Piazza, A. Policriti, and M. Simeoni. Modeling cellular behavior with hybrid automata: bisimulation and collapsing, 2003.

[3] M. Antoniotti, N. Ramakrishnan, and B. Mishra. Goalie, a common lisp application to discover kripke models: Redescribing biological processes from time-course data. In *International Lisp Conference, ILC 2005*, 2005.

[4] S. Arimoto. An algorithm for calculating the capacity of an arbitrary discrete memoryless channel. IT-18:14–20, 1972.

[5] M. Ashburner, CA Ball, JA Blake, D. Botstein, H. Butler, JM Cherry, AP Davis, K. Dolinski, SS Dwight, JT Eppig, et al. Gene ontology: tool for the unification of biology. the gene ontology consortium. *Nat Genet*, 25(1):25–9, 2000.

[6] R. Blahut. Computation of channel capacity and rate distortion functions. IT-18:460–473, 1972.

[7] Z. Bozdech, M. Llinas, BL Pulliam, ED Wong, J. Zhu, and JL DeRisi. The transcriptome of the intraerythrocytic developmental cycle of. *Plasmodium falciparum. PLoS Biol*, 1, 2003.

[8] A. Casagrande, K. Casey, C. Falch, R. Piazza1, B. Rupert, G. Vizzotto, and B. Mishra. Translating time-course gene expression profiles into semi-algebraic hybrid automata via dimensionality reduction? *Submitted*, December 2007.

[9] E. M. Clarke, O. Grunberg, and D. A. Peled. *Model Checking*. MIT Press, 1999.

[10] Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. John Wiley and Sons, Inc., 1991.

[11] I. Csiszár and G. Tusnady. Information geometry and alternating minimization procedures. *Statistics and Decisions*, Supplementary Issue 1:205.237, 1984.

[12] M.B. Eisen, P.T. Spellman, P.O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proc Natl Acad Sci U S A*, 95(25):14863–8, 1998.

[13] Message Passing Interface Forum. MPI: A message-passing interface standard. Technical Report UT-CS-94-230, 1994.

[14] Chris Fraley and Adrian E. Raftery. How many clusters? which clustering method? answers via model-based cluster analysis. *The Computer Journal*, 41(8):578–588, 1998.

[15] Thomas Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science (LICS '96)*, pages 278–292, New Brunswick, New Jersey, 1996.

[16] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[17] S. Kleinberg, K. Casey, and B. Mishra. Redescription in the real world: an ontology-based tool for discovery in large scale biological systems. *Submitted to 2007 International Conference on Life System Modeling and Simulation*.

[18] J. B. Macqueen. Some methods of classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathemtical Statistics and Probability*, pages 281–297, 1967.

[19] S. Madeira and A. Oliveira. Biclustering algorithms for biological data analysis: a survey, 2004.

[20] Marina Meila. Comparing clusterings. Technical report, 2002.

[21] Dan Pelleg and Andrew Moore. X-means: Extending K-means with efficient estimation of the number of clusters. In *Proc. 17th International Conf. on Machine Learning*, pages 727–734. Morgan Kaufmann, San Francisco, CA, 2000.

[22] Carla Piazza, Marco Antoniotti, Venkatesh Mysore, Alberto Policriti, Franz Winkler, and Bud Mishra. Algorithmic algebraic model checking i: Challenges from systems biology. In *CAV*, pages 5–19, 2005.

[23] K. Rose. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems, 1998.

[24] Gideon Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

[25] E. Segal, M. Shapira, A. Regev, D. Pe'er, 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):166–176, June 2003.

[26] Claude E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–, july, october 1948.

[27] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[28] Noam Slonim, Gurinder S. Atwal, Gasper Tkacik, and William Bialek. Estimating mutual information and multi–information in large networks, Feb 2005.

[29] Noam Slonim, Gurinder Singh Atwal, Gasper Tkacik, and William Bialek. Information based clustering, 2005.

[30] Susanne Still and William Bialek. How many clusters? an information theoretic perspective, May 2004.

[31] S. Tadepalli, N. Ramakrishnan, M. Antoniotti, K. Casey, and B. Mishra. On a mathematical method for reverse engineering dynamic models from gene expression data: Application to the yeast metabolic cycle. 2007.

[32] Naftali Tishby, Fernando C. Pereira, and William Bialek. The information bottleneck method, 2000.

[33] BP Tu, A Kudlicki, M Rowicka, and SL McKnight. Logic of the yeast metabolic cycle: temporal compartmentalization of cellular processes. *Science*, 310(5751):1152–8, 2005 Nov 18.

[34] Ya Zhang, Hongyuan Zha, and Chao-Hisen Chu. A time-series biclustering algorithm for revealing co-regulated genes. In *ITCC '05: Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05) - Volume I*, pages 32–37, Washington, DC, USA, 2005. IEEE Computer Society.