

Reverse engineering dynamic temporal models of biological processes and their relationships

Naren Ramakrishnan^a, Satish Tadepalli^a, Layne T. Watson^{a,b}, Richard F. Helm^c, Marco Antoniotti^f, and Bud Mishra^{d,e,1}

^aDepartment of Computer Science, Virginia Tech, Blacksburg, VA 24061; ^bDepartment of Mathematics, Virginia Tech, Blacksburg, VA 24061; ^cDepartment of Biochemistry, Virginia Tech, Blacksburg, VA 24061; ^dCourant Institute of Mathematical Sciences, New York University, New York, NY 10003; ^eNew York University (NYU) School of Medicine, New York, NY 10003; and ^fDipartimento di Informatica, Sistemistica e Comunicazione, Università degli Studi di Milano Bicocca, U14 Viale Sarca 336, I-20126 Milan, Italy

Communicated by Michael H. Wigler, Cold Spring Harbor Laboratory, Cold Spring Harbor, NY, May 10, 2010 (received for review June 3, 2009)

Biological processes such as circadian rhythms, cell division, metabolism, and development occur as ordered sequences of events. The synchronization of these coordinated events is essential for proper cell function, and hence the determination of critical time points in biological processes is an important component of all biological investigations. In particular, such critical time points establish logical ordering constraints on subprocesses, impose prerequisites on temporal regulation and spatial compartmentalization, and situate dynamic reorganization of functional elements in preparation for subsequent stages. Thus, building temporal phenomenological representations of biological processes from genome-wide datasets is relevant in formulating biological hypotheses on: how processes are mechanistically regulated; how the regulations vary on an evolutionary scale, and how their inadvertent dysregulation leads to a diseased state or fatality. This paper presents a general framework (GOALIE) to reconstruct temporal models of cellular processes from time-course gene expression data. We mathematically formulate the problem as one of optimally segmenting datasets into a succession of “informative” windows such that time points within a window expose concerted clusters of gene action whereas time points straddling window boundaries constitute points of significant restructuring. We illustrate here how GOALIE successfully brings out the interplay between multiple yeast processes, inferred from combined experimental datasets for the cell cycle and the metabolic cycle.

model building and model-checking | temporal data analysis | yeast cell cycle | yeast metabolic cycle | Kripke structures

Cells and organisms can be viewed as progressing through sequences of states, as a result of discrete mechanisms. Defining these states and identifying the underlying mechanisms are critical to how we understand biological processes and how we may treat metabolic and developmental disorders. Central to such analysis tools are algorithms for time series analysis using temporal logic formalisms that were originally developed with engineering and computer and systems sciences applications in mind (1, 2, 3).

The yeast species *Saccharomyces cerevisiae*, which has been researched extensively to understand the biology of eukaryotic microorganisms, is a good model organism to illustrate the ideas in this paper. To understand the systems biology of yeast, one may study temporal expression profiles of genes involved in a particular function—for instance, cellular (4) division or metabolism (5)—and create models of the state space dynamics in terms of labeled states and state transition relations. An illustration of this process is shown in Fig. 1. A yeast cell cycle (YCC) model can be created using data generated by Spellman et al. (6) and similarly, a yeast metabolic cycle (YMC) model can be created by combining data generated separately by two other research groups: Tu et al. (5), Klevecz et al. (7). These labeled state transition models are shown in the two insets in Fig. 1; formally, they can be viewed as Kripke structures (8), with atomic propositional labels corresponding to the Gene Ontology (GO) functional categories,

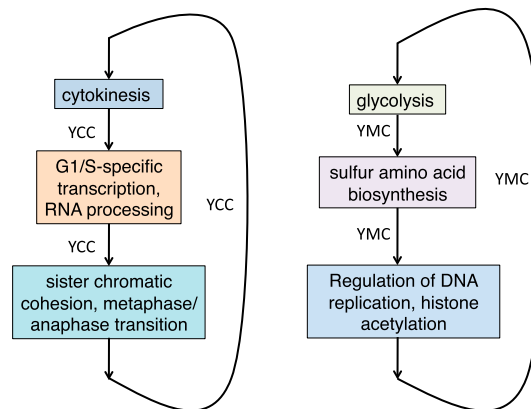


Fig. 1. Temporal process models reconstructed from segmentation algorithm. States are identified through the segmentation algorithm and edges are labeled by the experimental conditions under which the transitions are observed. (left) YCC. (right) YMC.

thus enabling temporal logic model-checking to extract complex global properties of these modules. For instance, we learn from the Kripke structure of the cell cycle that for cytokinesis to lead to DNA replication, the cell size must have enlarged sufficiently for division.

A key goal of this type of analysis is to be able to formulate models without preexisting hypotheses. For instance, how would the system behave when subjected to multiple perturbations? As an illustration, in Fig. 2, we computationally integrate data from the distinct YCC and YMC experiments along with data from other perturbations (e.g., by hydrogen peroxide (HP) or menadione (MD) treatments) into a more complex combined model. Such integration is possible even though the data sources for each experiment and perturbation were gathered independently. The combined model, created by this metaanalysis, reveals insightful and complex temporal properties of the combined system, not visible in the individual component models: for instance, the exit from cell cycle under HP perturbation is inferred as fundamentally different from that under MD treatment, in that under the latter the cells complete one full cycle before being arrested.

To create Kripke structure models as shown in Fig. 2, we require algorithms to extract states and state transitions from the data and subsequently, to label the states. Our contribution here is of a methodological nature: we devise a mathematically

Author contributions: N.R., R.F.H., and B.M. designed research; N.R., S.T., L.T.W., R.F.H., M.A., and B.M. performed research; L.T.W. and B.M. contributed new reagents/analytic tools; N.R., S.T., R.F.H., and M.A. analyzed data; and N.R., R.F.H., and B.M. wrote the paper.

The authors declare no conflict of interest.

Freely available online through the PNAS open access option.

¹To whom correspondence should be addressed at: Courant Institute of Mathematical Sciences, 251 Mercer Street, New York, NY 10012. E-mail: mishra@nyu.edu.

This article contains supporting information online at www.pnas.org/lookup/suppl/doi:10.1073/pnas.1006283107/-DCSupplemental.

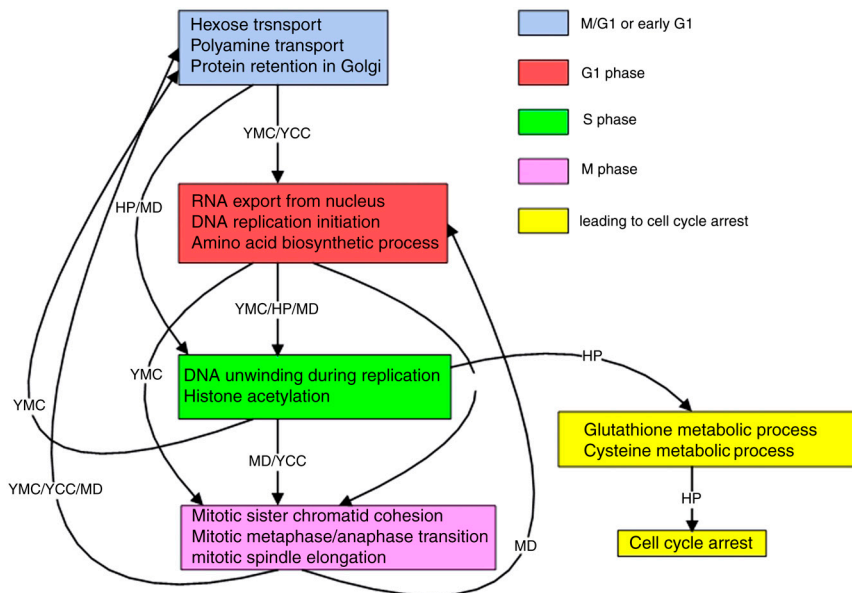


Fig. 2. Combined temporal process model of the YCC, YMC, and exposure of yeast cells to HP and MD treatment.

rigorous method (GOALIE) to reconstruct temporal models of cellular processes from time-course gene expression data.

GOALIE: A Mathematical Method to Infer Temporal Models

GOALIE aims to assist an experimentalist to track which genes are under coordinated temporal regulation, how their expression persists and dynamically varies over time, hence revealing insight into the progression of events constituting a given process.

As studied here, time-course analysis (9) has become an important tool for the study of organism development, disease progression, and cyclical biological processes, e.g., the cell cycle (6), metabolic cycle (5), and even entire life cycles (10). The growing affordability of transcriptional profiling screens has fostered the generation of numerous time series datasets. When the number of time points is sufficiently large, researchers have utilized continuous representations to smooth out noise (11), applied hidden Markov models to guide clustering (12), and used static measurements to “fill in the gaps” in the time series data (13). When the number of time points is small, researchers have studied the role played by sampling rates (14) and proposed the use of model profiles (15) to guide clustering. Recently, researchers have begun to characterize periodicity in transcriptional profiles (16), quantify timing differences in gene expression (17, 18), to extract binary signals embedded in the data (19), and to reconstruct regulatory relationships (20).

One of the attractions of time series analysis is its promise to reveal temporal relationships underlying biological processes (21): which process occurs before what, what are the “check-points” that must be satisfied (and when), and whether there can be alternative pathways of time series progression. There are many temporal modeling formalisms available: linear time logic uses a single time line from start to end whereas branching time logics allow multiple possibilities. In this paper, we primarily use linear time logics for modeling individual experimental conditions (stresses) yielding Gantt chart representations. The interplay between the stresses and potential branching behavior is then inferred indirectly through concurrent notations such as Kripke diagrams. A direction of future work is to directly model branching time behavior using data mining techniques.

Given gene expression time-course data, we identify segments of the time course as follows: Within a segment, genes organize into groups depicting concerted behavior but across segments there is a significant “regrouping” of genes. Although such

analysis can be conducted by tracking individual genes whose function is known, in GOALIE we desire to automatically mine, in an unsupervised manner, temporal relationships involving groups of genes, which are not a priori defined. Similarly motivated research can be found in, e.g., ref. 22, which models the activity level of genes as a mixture model of activity levels of processes but these works are not directly concerned with revealing the temporal coordination of processes—the focus of this paper. Biclustering algorithms such as presented in refs. 23 and 24 can reveal patterns that preserve order of the time course, but such biclusters do not yield a partition of the entire time course as is done here. We focus on deriving a complete segmentation from which we can construct a finite state model where the states (nodes) correspond to the segments and transitions (edges) denote observed state-state boundaries. Furthermore, propositions that hold true in each state (e.g., the biological processes enriched in the corresponding segment) can be used to label the corresponding node in the model. Models from multiple stresses are then combined by coalescing nodes and factoring transitions.

Information-Theoretic Framework

Given multiple vectors of measurements $\mathcal{G} = \{\mathbf{g}_1, \mathbf{g}_2, \dots, \mathbf{g}_N\}$, where each \mathbf{g}_i is a time series over $\mathcal{T} = \{t_1, t_2, \dots, t_l\}$, the problem of segmentation is to find a sequence of segments, $(w_{t_1}^a, w_{t_{a+1}}^b, \dots, w_{t_l}^c)$ where each segment $w_{t_s}^e$, $t_s \leq t_e$, is a set of consecutive time points beginning at time point t_s and ending at time point t_e . The segment boundaries are identified such that, were we to cluster genes in neighboring windows separately, we would find highly dissimilar sets of clusters. Consider the case of segmentation into just two windows: i.e., we seek to find segments $w_{t_a}^b$ and $w_{t_{b+1}}^c$, with r clusters in the window $w_{t_a}^b$ and c clusters in the window $w_{t_{b+1}}^c$. Let α and β represent the cluster random variables for the windows $w_{t_a}^b$ and $w_{t_{b+1}}^c$ respectively. (The set of possible values for α is $\{1, \dots, r\}$ and for β is $\{1, \dots, c\}$.)

We begin by measuring the similarity of the clusters across the windows using a $r \times c$ contingency table. Entry n_{ij} in the $(i, j)^{\text{th}}$ cell of the table represents the overlap between the genes clustered together in cluster i of window $w_{t_a}^b$ and in cluster j of window $w_{t_{b+1}}^c$. The sizes of the clusters in $w_{t_a}^b$ are given by the column-wise sums across each row: $n_{i\cdot} = \sum_j n_{ij}$, while the sizes of clusters in $w_{t_{b+1}}^c$ are given by row-wise sums down each column: $n_{\cdot j} = \sum_i n_{ij}$.

Interpreting each row and column as a probability distribution, ideally highly dissimilar clusters result in a total of $(r + c)$ uniform

distributions across the rows and columns of the contingency table. To capture the deviation of these distributions from the uniform distribution, we define r random variables $R_i, i = 1, \dots, r$ occurring with probability $p_{R_i}(j) = \frac{n_{ij}}{n_i}$ corresponding to each row. Similarly, we define c random variables $C_j, j = 1, \dots, c$ occurring with probability $p_{C_j}(i) = \frac{n_{ij}}{n_j}$ corresponding to each column. We capture the deviation of these distributions from the uniform distributions over the rows ($U(\frac{1}{r})$) and columns ($U(\frac{1}{c})$) by

$$\frac{1}{r} \sum_{i=1}^r D_{\text{KL}}\left(p_{R_i} \parallel U\left(\frac{1}{r}\right)\right) + \frac{1}{c} \sum_{j=1}^c D_{\text{KL}}\left(p_{C_j} \parallel U\left(\frac{1}{c}\right)\right), \quad [1]$$

where $D_{\text{KL}}(p \parallel q) = \sum_x p(x) \log_2 \frac{p(x)}{q(x)}$ is the Kullback-Leibler (KL) divergence between two probability distributions $p(x)$ and $q(x)$. We can thus cluster the adjacent windows using this objective function, minimizing it in order to yield highly dissimilar clusters across the windows. Since the KL divergence of any distribution with respect to the uniform distribution differs from its negative entropy by a constant (when the sizes of the supports of the distributions are fixed), [1] can be equivalently expressed as

$$\begin{aligned} \mathcal{F} &= -\frac{1}{r} \sum_{i=1}^r H(R_i) - \frac{1}{c} \sum_{j=1}^c H(C_j), \\ &= -\frac{1}{r} \sum_{i=1}^r H(\beta | \alpha = i) - \frac{1}{c} \sum_{j=1}^c H(\alpha | \beta = j). \end{aligned} \quad [2]$$

Here $H(X)$ (resp. $H(X|Y)$) denotes the entropy (resp. relative entropy) of a probability mass function $p(x)$ (resp. $p(x|y)$) for X (resp. X relative to Y). Thus the function \mathcal{F} captures the mutual information between the clusterings in adjacent windows.

Our goal is to minimize \mathcal{F} and obtain clusters that are local within each segment (similar to a k -means algorithm) but have high dissimilarity when compared with clusterings from the neighboring segment. We achieve this by parameterizing \mathcal{F} in terms of cluster prototypes, defining the cluster random variables to capture locality in their respective spaces, and optimizing \mathcal{F} using an augmented Lagrangian algorithm (see *SI Appendix* for details).

To identify the segments we employ a dynamic programming algorithm. Using minimum and maximum segment length constraints, we consider all possible “tilings” of the time course where every pair of neighboring tiles reduces to the problem above, i.e., where the evaluation consists of applying our clustering framework and determining the minimized value of \mathcal{F} . These objective function values are then summed over an entire segmentation and used to evaluate one segmentation over another.

Computationally, this reduces to a shortest path algorithm where each edge length is given by the minimized value of \mathcal{F} .

Results

Specific strains of *Saccharomyces cerevisiae* have been shown to have two robust biological cycles occurring simultaneously, e.g., the metabolic and cell cycles (25, 26). The GOALIE framework is validated through analysis of five yeast gene expression datasets, including two YMC time courses involving two different strains grown under two different conditions (YMC1: CEN.PK122 diploid strain, glucose-limited cultures (5) and YMC2: IFO 0233 diploid strain, not glucose limited (7)), a YCC dataset after release from α -factor synchronization (YCC: DBY8724 strain (6)), and observations of the cell cycle under treatment of (HP (27) and (MD (27)).

Yeast Cell Cycle. We computed the optimal segmentation for the YCC α -factor synchronization experiment of Spellman et al. (6) using GOALIE’s dynamic programming algorithm. This dataset comprises two cycles, one of which is explained in detail in Fig. 3 and both cycles are summarized in the complete segmentation (*SI Appendix*). To understand the temporal nature of the underlying dataset, in Fig. 3 we label each window with only functions from the cluster whose mean expression peaks during the window. We make several qualitative observations from the segmentation. First, from Fig. 3, observe how clusters within each window offer significant enrichments of biological processes (*contingency tables in the first row*) whereas there is significant regrouping of genes across neighboring windows (*contingency tables in the second row*). Second, GOALIE’s segmentation brings out the cyclic nature of the dataset—alternating M/G1, {G1,S}, {G2,M} phases—without explicit instruction. By studying the processes enriched in each segment of Fig. 3, the careful coordination of the cell cycle is easily seen. As stated in (6), the YCC time-course data spans approximately two points each for phases M/G1, G1, and S and spans only one time point for the G2 phase. Because our minimum window length is three (set so that we recover significant clusterings and regroupings), we cannot precisely resolve these short-lived phases with this dataset. A possible approach is to use continuous representations such as spline fits to gain greater resolution of data sampling (15). Nevertheless, the key events occurring in these segments are retrieved with high specificity ($p < 10^{-7}$).

Yeast Metabolic Cycle. While the YCC has been well studied, the timing relationships in the YMC have only recently become elucidated. For instance, a main result of ref. 5 is the existence of three key clusters of expression patterns that oscillate coordinatively through the metabolic cycle phases, influenced by careful

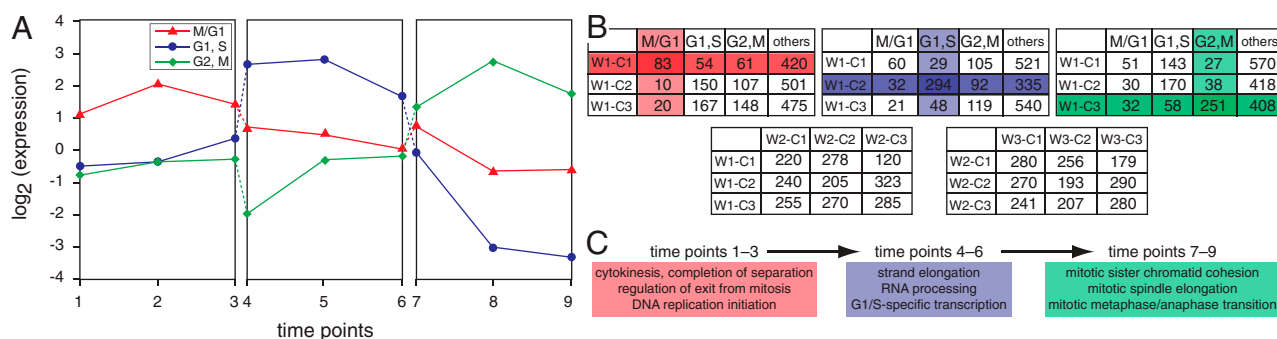
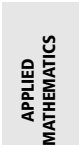


Fig. 3. Preview of results from segmenting the YCC dataset. Only one cycle is shown here. The YCC involves the staged coordination of several phases (M/G1, time points [1–3]; G1/S, time points [4–6]; and G2/M, time points [7–9]). (A) Mean expression profiles for each group of genes depict the changing emphasis across the three phases. Contingency tables capture the concerted grouping of genes within segments (*B, first row*) as well as the regroupings between segments (*B, second row*). Observe that the contingency tables in the *first row* involve significant enrichments whereas the tables in the *second row* approximate a uniform distribution. Gantt chart views (*C*) depict the temporal coordination of biological processes underlying the dataset. Only some of the enriched functions are displayed, for lack of space.



Ramakrishnan et al.

dual cell and its decision to divide? Can we use gene knockouts and/or growth condition modifications to separate the YMC and YCC so that they are independent of one another?

For example, our investigation of the transcriptional profiling associated with peroxide stress identified a time segment that corresponded to an “intermediate stage” (Fig. 5, *Segment 3*) where the yeast cells were recovering from peroxide stress. The GO categories enriched in this segment were related to core metabolic processes (ethanol, TCA, glycogen), sulfur metabolism, and inositol lipid-mediated signaling, as well as chromatin silencing and nuclear pore organization/biogenesis. While sulfur metabolism can be associated directly with the oxidative stress response, the linkage to inositol lipid-mediated signaling genes and chromatin silencing is a bit more remote. Nevertheless, our tools bring out the nature of temporal “hardwiring” manifest in biological processes. In particular they open up questions related to whether it would be possible to manipulate the system to adopt an aberrant cell state or make it proceed along a desired temporal order. For instance, the identification of well-defined transcriptional states such as found in Segment III of the peroxide treatment suggests that at this stage in the cell growth regime it may be possible to force the organism to adopt aberrant states. For example, exit from peroxide treatment results in entry into the G2/M state. What would be the effect of adding alpha factor to the growth medium directly after release from peroxide stress? Would the cells continue through the cell cycle once before entering into G1, or move directly to G1?

Discussion

This work builds upon our prior research (2, 3) to make two key contributions. First, it provides successful inferences from multiple yeast time-course datasets, demonstrating the wide applicability of our information-theoretic methods. Second, unlike prior research, it focuses on teasing out relationships *across stresses* and summarizing process-level relationships in an integrated temporal model. In particular, we have uncovered the stages of peroxide stress response and situated them in relation to the YCC and metabolic cycle response.

Simply by extracting and analyzing the connections between the YMC and the YCC processes, which had remained latent

in published data, it seems possible to refine hypotheses involving biochemical process compatibility versus coordinated metabolic “bursts,” which are currently under intense investigation (25). Temporal analysis of existing data points could lead to a systematic way to generate and experimentally refute (hitherto) nonobvious hypotheses.

Methods

Datasets and Data Preprocessing. Our datasets came from a variety of sources (see *SI Appendix*). For each dataset we retained only genes that have an annotation in the GO biological process taxonomy (revision 4.205 of GO released on March 14, 2007), log transformed (base 10) their expression values and normalized them such that the mean expression of each gene across all time points is zero.

Dynamic Programming Algorithm for Optimal Segmentation. As described elsewhere (3), we apply a dynamic programming algorithm for segmenting the various time series. We used different settings for the numbers of clusters and different thresholds for minimum and maximum possible window lengths to search in the space of possible segmentations. Besides the number of clusters in each segment, and minimum/maximum constraints on window lengths, we parameterized the segmentation algorithm with a parameter λ that controls the sizes of the clusters in the resulting segmentations and can be adjusted to yield approximately equal cluster sizes (see *SI Appendix*). After the segmentation reveals windows and clusters of genes in each window, we perform functional enrichment over the selected sets of genes. A hypergeometric p -value is calculated for each GO biological process term, and an appropriate cutoff is chosen using false discovery rate q -level of 0.01 (28). The time bounded enrichments are summarized as Gantt charts as presented earlier. We employ various statistical tests to assess the sensitivity of the segmentation to variations in the number of clusters (see *SI Appendix*).

Inferring Temporal Coordination of Processes. We derive temporal process models from Gantt charts as follows: Given two neighboring segments, we assume that each of the processes enriched in the first segment precedes (i.e., has a state transition to) a process enriched in the second segment. We then find maximal sets of processes that are common across two or more datasets that obey the same precedence relationships.

ACKNOWLEDGMENTS. This work was supported in part by National Science Foundation Grants (ITR-0428344, CCF-0836649, CCF-0937133), and the Institute for Critical Technology and Applied Science (ICTAS) at Virginia Tech.

- Kleinberg S, Casey K, Mishra B (2008) Systems biology via redescription and ontologies (I): Finding phase changes with applications to malaria temporal data. *Systems and Synthetic Biology Journal* 1:197–205.
- Ramakrishnan N, Antoniotto M, Mishra B (2005) Reconstructing formal temporal logic models of cellular events using the GO process ontology. *Proceedings of the Eighth Annual Bio-Ontologies Meeting (ISMB (Intelligent Systems for Molecular Biology), Detroit MI)* p 2.
- Tadepalli S, Ramakrishnan N, Watson L, Mishra B, Helm R (2009) Simultaneously segmenting multiple gene expression time courses by analyzing cluster dynamics. *Journal of Bioinformatics and Computational Biology* 7:339–356.
- Chen K, et al. (2004) Integrative analysis of cell cycle control in budding yeast. *Mol Biol Cell* 15:3841–3862.
- Tu B, Kudlicki A, Rowicka M, McKnight S (2005) Logic of the yeast metabolic cycle: temporal compartmentalization of cellular processes. *Science* 310:1152–1158.
- Spellman P, et al. (1998) Comprehensive identification of cell cycle-regulated genes of the yeast *Saccharomyces cerevisiae* by microarray hybridization. *Mol Biol Cell* 9:3273–3297.
- Klevecz R, Bolen J, Forrest G, Murray DB (2004) A genomewide oscillation in transcription gates DNA replication and cell cycle. *Proc Natl Acad Sci USA* 101:1200–1205.
- Clarke E, Grumberg O, Peled D (1999) *Model Checking* (MIT Press, Boston, MA).
- Bar-Joseph Z (2004) Analyzing time series gene expression data. *Bioinformatics* 20:2493–2503.
- Lund J, et al. (2002) Transcriptional profile of aging in *C. elegans*. *Curr Biol* 12:1566–1573.
- Bar-Joseph Z, Gerber G, Gifford DK, Jaakkola T, Simon I (2003) Continuous representations of time-series gene expression data. *J Comput Biol* 10:341–356.
- Schliep A, Schonhuth A, Steinhoff C (2003) Using hidden Markov models to analyze gene expression time course data. *Bioinformatics* 19:i255–i263.
- Simon I, Siegfried Z, Ernst J, Bar-Joseph Z (2005) Combined static and dynamic analysis for determining the quality of time-series expression profiles. *Nat Biotechnol* 23:1503–1508.
- Singh R, Palmer N, Gifford D, Berger B, Bar-Joseph Z (2005) Active learning for sampling in time-series experiments with application to gene expression analysis. *Proceedings of the Twenty-Second International Conference on Machine Learning (ICML'05)* (Association for Computing Machinery (ACM), New York), pp 832–839.
- Ernst J, Nau GJ, Bar-Joseph Z (2005) Clustering short time series gene expression data. *Bioinformatics* 21:i159–i168.
- Kudlicki A, Rowicka M, Otwinowski Z (2007) SCEPTRANS: An online tool for analyzing periodic transcription in yeast. *Bioinformatics* 23:1559–1561.
- Rowicka M, Kudlicki A, Tu B, Otwinowski Z (2007) High-resolution timing of cell cycle-regulated gene expression. *Proc Natl Acad Sci USA* 104:16892–16897.
- Yoneya T, Mamitsuka H (2007) A hidden Markov model-based approach for identifying timing differences in gene expression under different experimental factors. *Bioinformatics* 23:842–849.
- Sahoo D, Dill D, Tibshirani R, Plevritis S (2007) Extracting binary signals from microarray time-course data. *Nucleic Acids Res* 35:3705–3712.
- Shi Y, Mitchell T, Bar-Joseph Z (2007) Inferring pairwise regulatory relationships from multiple time series datasets. *Bioinformatics* 23:755–763.
- de Lichtenberg U, Jensen L, Brunak S, Bork P (2005) Dynamic complex formation during the yeast cell cycle. *Science* 307:724–727.
- Segal E, Battle A, Koller D (2003) Decomposing gene expression into cellular processes. *Proceedings of the Pacific Symposium on Biocomputing (PSB'03)* (World Scientific Press, Singapore), pp 89–100.
- Madeira SC, Oliveira AL (2009) A polynomial time biclustering algorithm for finding approximate expression patterns in gene expression time series. *Algorithms Mol Biol* 4:8.
- Supper J, Strauch M, Wanke D, Harter K, Zell A (2007) EDISA: Extracting biclusters from multiple time-series of gene expression profiles. *BMC Bioinformatics* 8:334.
- Futcher B (2006) Metabolic cycle, cell cycle, and the finishing kick to start. *Genome Biol* 7:107.
- Palkova Z, Vachova L (2006) Life within a community: benefit to yeast long-term survival. *FEMS Microbiol Rev* 30:806–824.
- Shapira M, Segal E, Botstein D (2004) Disruption of yeast forkhead-associated cell cycle transcription by oxidative stress. *Mol Biol Cell* 15:5659–5669.
- Storey J, Tibshirani R (2003) Statistical significance for genomewide studies. *Proc Natl Acad Sci USA* 100:9440–9445.

Supplementary Information to accompany Reverse Engineering Dynamic Temporal Models of Biological Processes and their Relationships

N. Ramakrishnan *et al.*

This document provides supplementary details to accompany the manuscript, in particular it presents a complete description of the algorithm and an outline of our experimental methodology. Complete Gantt charts for each of the stresses analyzed is also presented here.

1 Algorithm Description

As discussed in the main text, the ‘unit’ of our analysis is to simultaneously cluster two neighboring windows such that the clusters are maximally dissimilar in terms of the related samples assigned to them. This unit is then composed to yield a tiling of the entire time series, or a segmentation.

1.1 Formulation

Given a gene vector \mathbf{g}_k , let its projection onto the ‘left’ window $w_{t_a}^{t_b}$ be referred to as \mathbf{x}_k , and its projection onto the ‘right’ window $w_{t_{b+1}}^{t_c}$ be referred to as \mathbf{y}_k . Recall that sets of such projections are clustered separately such that the clusters are maximally dissimilar. Let r and c be the number of clusters for \mathbf{x} and \mathbf{y} vectors, which results in an $r \times c$ contingency table. Let $\mathbf{m}_i^{(x)}$ be the prototype vector for the i^{th} cluster of the \mathbf{x} vectors. The random variable $V^{(\mathbf{x}_k)}$ denotes the assignment of the data vector \mathbf{x}_k to the clusters: the probability of \mathbf{x}_k being assigned to cluster i is given by $P(V^{(\mathbf{x}_k)} = i) = v_i^{(\mathbf{x}_k)}$, where $\sum_{i=1}^r v_i^{(\mathbf{x}_k)} = 1$. We refer to the probabilities $v_i^{(\mathbf{x}_k)}$ as cluster membership indicator variables. Similar cluster prototypes $\mathbf{m}_j^{(y)}$, random variables $V^{(\mathbf{y}_k)}$, and cluster indicator variables $v_j^{(\mathbf{y}_k)}$ are defined for \mathbf{y} vectors as well. We denote the probability mass function associated with each $V^{(\mathbf{x}_k)}$ as $p_{V^{(\mathbf{x}_k)}}$, and with each $V^{(\mathbf{y}_k)}$ as $p_{V^{(\mathbf{y}_k)}}$. Then the contingency table counts can be calculated as $n_{ij} = \sum_{k=1}^N v_i^{(\mathbf{x}_k)} v_j^{(\mathbf{y}_k)}$. In *hard clustering* algorithms, like the traditional k -means, each data sample is assigned to the nearest cluster with a probability of 1. However, calculating n_{ij} using hard memberships renders the function \mathcal{F} (see main text) nondifferentiable at certain points, as a result of which, we cannot leverage classical numerical optimization algorithms to minimize \mathcal{F} . To avoid this problem, cluster indicator variables are typically treated as continuous real variables making \mathcal{F} a smooth function that is continuously differentiable and assigning a nonzero cluster membership probability for each data sample, i.e., $v_i^{(\mathbf{x}_k)}, v_j^{(\mathbf{y}_k)} \in (0, 1)$.

1.2 Smoothing Cluster Probabilities

There are many ways of smoothing, one approach being the use of a Gaussian kernel between the vector and the cluster prototype. We present a novel derivation of this kernel that explains how the error in the

smoothing can be explicitly controlled and also suggests other formulations for smoothing. First, we define

$$\gamma_{(i,i')}(\mathbf{x}_k) = \frac{||\mathbf{x}_k - \mathbf{m}_{i'}^{(x)}||^2 - ||\mathbf{x}_k - \mathbf{m}_i^{(x)}||^2}{D}, 1 \leq i, i' \leq r,$$

where $D = \max_{k,k'} ||\mathbf{x}_k - \mathbf{x}_{k'}||^2$, $1 \leq k, k' \leq N$ is the pointset diameter.

The non-normalized cluster assignment probabilities are given by

$$\hat{v}_i^{\mathbf{x}_k} = \exp(\rho(\min_{i'} \gamma_{(i,i')}(\mathbf{x}_k)))$$

and the normalized probabilities are then given by

$$v_i^{(\mathbf{x}_k)} = \frac{\hat{v}_i^{\mathbf{x}_k}}{\sum_{i'} \hat{v}_{i'}^{\mathbf{x}_k}} \quad (1)$$

A well known approximation to $\min_{i'} \gamma_{(i,i')}(\mathbf{x}_k)$ is the Kreisselmeier-Steinhaus (KS) envelope function [1, 4, 11] given by

$$KS_i(\mathbf{x}_k) = \frac{-1}{\rho} \ln \left[\sum_{i'=1}^r \exp(-\rho \gamma_{(i,i')}(\mathbf{x}_k)) \right],$$

where $\rho \gg 0$. The KS function is a smooth function that is infinitely differentiable. Using this the cluster membership indicators are redefined as:

$$v_i^{(\mathbf{x}_k)} = \frac{\exp[\rho KS_i(\mathbf{x}_k)]}{\sum_{i'=1}^r \exp[\rho KS_{i'}(\mathbf{x}_k)]} = \frac{\exp(-\frac{\rho}{D} ||x_k - m_i||^2)}{\sum_{i'=1}^r \exp(-\frac{\rho}{D} ||x_k - m_{i'}||^2)} \quad (2)$$

The cluster memberships for the “right” window, $v_j^{(\mathbf{y}_k)}$, are also smoothed similarly.

Note that ρ/D is the width of the Gaussian kernel used for approximation, however the KS-function helps tease out how the width must be set in order to achieve a certain quality of approximation. Furthermore, D is completely determined by the data but ρ is a user-settable parameter, and precisely what we can tune. The KS-function provides bounds on the error with which it approximates $\min_{i'} \gamma_{(i,i')}(\mathbf{x}_k)$, given as follows

$$\min_{i'} \gamma_{(i,i')}(\mathbf{x}_k) - \frac{\ln(N)}{\rho} \leq KS_i(\mathbf{x}_k) \leq \min_{i'} \gamma_{(i,i')}(\mathbf{x}_k)$$

The above inequality shows that the value of ρ can be determined for a given error precision. Furthermore, we can use any other rapidly decaying function to assign the probabilities (i.e. in place of the exponential) and the KS-function would again help us smooth the resulting assignments, but will yield assignment probabilities that are quite different from the traditional Gaussian kernel. In this sense, the KS-approximation is a versatile approach to smooth a variety of functions.

1.3 Objective Function and Regularization

As discussed in the main text, the objective function can be written as:

$$\begin{aligned} \mathcal{F} &= -\frac{1}{r} \sum_{i=1}^r H(R_i) - \frac{1}{c} \sum_{j=1}^c H(C_j), \\ &= -\frac{1}{r} \sum_{i=1}^r H(\beta|\alpha = i) - \frac{1}{c} \sum_{j=1}^c H(\alpha|\beta = j) \end{aligned}$$

where the entropy terms derive from KL-divergences w.r.t. the uniform distribution. Minimizing the function \mathcal{F} should ideally yield clusters that are independent across windows and local within each window. However, using smooth cluster prototypes gives rise to an alternative minimum solution where each data sample is assigned with uniform probability to each cluster. For example, recall the 3×3 uniform contingency table example, where each of 18 samples can be assigned to 3 row clusters and 3 column clusters with probability $[1/3, 1/3, 1/3]$ and the estimate of the count matrix from these soft counts would still be uniform in each cell ($\sum_k v_i^{(\mathbf{x}_k)} v_j^{(\mathbf{y}_k)} = 2$). To avoid degenerate solutions such as these, we require maximum deviation of individual data vector probabilities ($v_i^{(\mathbf{x}_k)}$ and $v_j^{(\mathbf{y}_k)}$) from the uniform distribution over the number of clusters. This leads to the regularized objective function:

$$\begin{aligned} \mathcal{F} = & \frac{\lambda}{r} \sum_{i=1}^r D_{KL} (p_{R_i} || U(\frac{1}{c})) + \frac{\lambda}{c} \sum_{j=1}^c D_{KL} (p_{C_j} || U(\frac{1}{r})) \\ & - \frac{1}{N} \sum_{k=1}^N D_{KL} (p_{V^{(\mathbf{x}_k)}} || U(\frac{1}{r})) - \frac{1}{N} \sum_{k=1}^N D_{KL} (p_{V^{(\mathbf{y}_k)}} || U(\frac{1}{c})), \end{aligned} \quad (3)$$

where λ is the weight, set to a value greater than 1, to give more emphasis to minimizing the row and column distributions. This also enforces equal cluster sizes.

The role of λ is to enforce a ‘balancing constraint’ on the clusters (i.e., approximately equal cluster sizes) and to prevent samples from being assigned to multiple clusters. Other approaches have focused on explicitly capturing these aspects by an objective function but here we intend λ to be a regularization parameter, intended to avoid degenerate solutions. Hence the exact value of λ is not as crucial as the regime in which we conduct the optimization. In order to adjust λ , we vary its value over a range (typically $[1, 2]$ in small step sizes). Based on experimentation, the cluster assignments of most gene vectors (more than 90%) do not change after a particular value of λ and we use this criterion to set λ . All the terms in the definition of \mathcal{F} above can be calculated in terms of the smoothed cluster membership probabilities $v_i^{(\mathbf{x}_k)}$ and $v_j^{(\mathbf{y}_k)}$, which are in turn calculated in terms of the cluster prototypes $\mathbf{m}_i^{(x)}$ and $\mathbf{m}_j^{(y)}$. Thus the objective function \mathcal{F} is effectively parametrized in terms of the cluster prototypes, and the problem of finding independent clusters now reduces to finding the cluster prototypes that optimize the objective function.

1.4 Optimization

The gradient of \mathcal{F} with respect to the prototypes $\mathbf{m}_i^{(x)}$ is given by

$$\begin{aligned} \nabla_{\mathbf{m}_i^{(x)}} \mathcal{F} = & \frac{1}{\ln(2)} \sum_{i'=1}^r \sum_{k=1}^N \left(\frac{\lambda}{r} \left\{ \sum_{j=1}^c \left[1 + \ln \left(\frac{\sum_{k'=1}^N v_{i'}^{(\mathbf{x}_{k'})} v_j^{(\mathbf{y}_{k'})}}{\sum_{k'=1}^N v_{i'}^{(\mathbf{x}_{k'})}} / \frac{1}{c} \right) \right] \right. \right. \\ & \left. \left[\frac{v_j^{(\mathbf{y}_k)}}{\sum_{k'=1}^N v_{i'}^{(\mathbf{x}_{k'})}} - \frac{\sum_{k'=1}^N v_{i'}^{(\mathbf{x}_{k'})} v_j^{(\mathbf{y}_{k'})}}{\sum_{k'=1}^N (v_{i'}^{(\mathbf{x}_{k'})})^2} \right] \right\} \\ & - \frac{\lambda}{c} \left\{ \sum_{j=1}^c \left[1 + \ln \left(\frac{\sum_{k'=1}^N v_{i'}^{(\mathbf{x}_{k'})} v_j^{(\mathbf{y}_{k'})}}{\sum_{k'=1}^N v_j^{(\mathbf{y}_{k'})}} / \frac{1}{r} \right) \right] \left[\frac{v_j^{(\mathbf{y}_k)}}{\sum_{k'=1}^N v_j^{(\mathbf{y}_{k'})}} \right] \right\} \\ & \left. - \frac{1}{N} \left[1 + \ln \left(v_{i'}^{(\mathbf{x}_k)} / \frac{1}{r} \right) \right] \right) \nabla_{\mathbf{m}_i^{(x)}} v_{i'}^{(\mathbf{x}_k)}, \end{aligned} \quad (4)$$

input $\mathcal{T} = (t_1, t_2, \dots, t_l)$: Given time series data sequence.
input l_{min} : Minimum window length.
input l_{max} : Maximum window length.

Step 1: Define the set of windows starting from time point t_a , $S_{t_a} = \{w_{t_a}^{t_b} | l_{min} \leq t_b - t_a + 1 \leq l_{max}\}$.

Step 2: Construct a directed acyclic graph where each $w_{t_a}^{t_b}$ is a node and a directed edge exists from $w_{t_a}^{t_b}$ to the windows $w_{t_{b+1}}^{t_c} \in S_{t_{b+1}}$.

Step 3: Cluster the adjacent windows $w_{t_a}^{t_b}$ and $w_{t_{b+1}}^{t_c}$ by minimizing the objective function in Eq.(3). Let $\mathcal{F}_{w_{t_a}^{t_b}, w_{t_{b+1}}^{t_c}}$ be the final value of the objective function. Assign $\mathcal{F}_{w_{t_a}^{t_b}, w_{t_{b+1}}^{t_c}}$ as the edge weight between the nodes represented by $w_{t_a}^{t_b}$ and $w_{t_{b+1}}^{t_c}$.

Step 4: Let $E_{t_l} = \{w_{t_k}^{t_l} | l_{min} \leq t_l - t_k + 1 \leq l_{max}\}$ be the set of windows ending in the last time point t_l . For each window starting at the first time point, $w_{t_1}^{t_b} \in S_{t_1}$, calculate the minimum cost path to all $w_{t_k}^{t_l} \in E_{t_l}$.

Step 5: Calculate $D_{Seg} = \mathcal{F}_{\{w_{t_1}^{t_a}, w_{t_a}^{t_b}\}} + \mathcal{F}_{\{w_{t_b}^{t_c}, w_{t_c}^{t_d}\}} + \dots + \mathcal{F}_{\{w_{t_j}^{t_k}, w_{t_k}^{t_l}\}}$ for each shortest path in step 4.

Step 6: Return the path with minimum D_{Seg} .

Figure 1: Algorithm for segmenting a time series

where

$$\nabla_{\mathbf{m}_i^{(x)}} v_i^{(\mathbf{x}_k)} = \frac{2\rho(\mathbf{x}_k - \mathbf{m}_i^{(x)})}{D} v_i^{(\mathbf{x}_k)} (\delta_{i',i} + v_i^{(\mathbf{x}_k)}) \quad (5)$$

Here $\delta_{i',i}$ is the Kronecker delta. The index variables i , i' , and i'' are over the clusters in the \mathbf{x} vectors, j over the clusters in the \mathbf{y} vectors, and k and k' over the data vectors. The gradients with respect to the prototypes $\mathbf{m}_j^{(y)}$ are calculated analogously.

Optimization of \mathcal{F} is performed using the augmented Lagrangian algorithm with simple bound constraints on the cluster prototypes using the FORTRAN package LANCELOT [2]. The initial cluster prototypes are set using individual k -means clusters in each window. The augmented Lagrangian algorithm iteratively improves these initial prototypes till a local minimum of the objective function is attained.

1.5 Dynamic programming

We can now present the algorithm for tiling a given time course into a series of windows such that neighboring windows capture breakpoints of significant re-organization (see Fig. 1). Here, l_{min} and l_{max} are the minimum and maximum window lengths. We create all possible ‘tiles’ of the entire time course that satisfy these constraints and then evaluate every consecutive pair of them. The evaluation consists of applying our clustering framework and determining the minimized value of \mathcal{F} . These objective functions are then summed over an entire segmentation and used to evaluate one segmentation over another. Computationally, this reduces to a shortest path algorithm where each edge length is given by the minimized value of \mathcal{F} . Each optimization can be performed in a matter of a few seconds on a desktop computer so that the entire segmentation is computable in a few minutes.

2 Evaluation and Assessment

We evaluate our clusterings and segmentations in five ways: cluster stability, cluster reproducibility, functional enrichment, segmentation quality, and segmentation sensitivity. Cluster stability and cluster reproducibility are used to filter the patterns obtained by the segmentation algorithm whereas the other three criteria are used to evaluate the quality of segmentation.

We assess **cluster stability** using a bootstrap procedure to determine significance of genes brought together. Recall that each window except the first and last windows has two sets of clusters, one set independent with respect to the previous window and the other independent with respect to the next window. We are interested in the genes that are significantly clustered together in these two sets of clusters, as they represent the genes that are specific to the window under consideration. We calculate a contingency table between these two clusterings for each window (excluding the first and the last window) in which each cell represents the number of genes that are together across the two independent sets of clusters. We randomly sample 1000 pairs of clusterings within each window (with cluster sizes same as the two independent clusterings) and compute their contingency tables. By the central limit theorem, the distribution of counts in each cell of the table is approximately normal (also verified using a Shapiro-Wilk normality test with $p = 0.05$). We now evaluate each cell of the actual contingency table with respect to the corresponding random distribution and retain only those cells that have more genes than that observed at random with $p < 0.05$ (Bonferroni corrected with the number of cross clusters to account for multiple hypothesis testing). To ensure **reproducibility** of clusters, we retain only those genes in each significant cell of the contingency table that are together in more than 150 of the 200 clusterings (conducted with different initializations). For the first and last windows, which have only 100 randomly initialized clusterings, we retain those genes that are clustered together in more than 75 of the 100 clusterings. At this stage, for each segment we obtain a contingency table with significant cells representing the group of genes specific to the particular segment, which we can refer to as cross-cluster. We perform **functional enrichment** using the GO biological process ontology (since we are tracking biological processes) over each of these cross clusters. A hypergeometric p -value is calculated for each GO biological process term, and an appropriate cutoff is chosen using a false discovery rate (FDR) q -level of 0.01.

The **segmentation quality** is calculated as a partition distance [6] between the “true” segmentation (from the literature of the YMC and YCC) to the segmentations computed by our algorithm. We view each window as a set of time points so that a segmentation is a partition of time points. Given two segmentations S_1 and S_2 , whose windows are indexed by the variables $w_{t_a}^{t_b}$ and $z_{t_c}^{t_d}$ respectively, the partition distance is given by:

$$\begin{aligned}
 P_D = & - \sum_{w_{t_a}^{t_b} \in S_1} \sum_{z_{t_c}^{t_d} \in S_2} |w_{t_a}^{t_b} \cap z_{t_c}^{t_d}| \log_2 \frac{|w_{t_a}^{t_b} \cap z_{t_c}^{t_d}|}{|w_{t_a}^{t_b}|} \\
 & - \sum_{z_{t_c}^{t_d} \in S_2} \sum_{w_{t_a}^{t_b} \in S_1} |w_{t_a}^{t_b} \cap z_{t_c}^{t_d}| \log_2 \frac{|w_{t_a}^{t_b} \cap z_{t_c}^{t_d}|}{|z_{t_c}^{t_d}|}.
 \end{aligned} \tag{6}$$

The **segmentation sensitivity** to variations in the number of clusters is calculated as the average of the ratios of KL-divergences between the segments to the maximum possible KL divergence between those segments. This latter figure is easy to compute as a function of the number of clusters, which is considered uniform throughout the segmentation. Suppose we have $|S|$ windows in a given segmentation $S = \{w_{t_1}^{t_a}, w_{t_{a+1}}^{t_b}, \dots, w_{t_{j+1}}^{t_k}, w_{t_{k+1}}^{t_l}\}$ with c clusters in each window. Let \mathcal{F}_{max} be the objective function

value for the maximally similar clustering (the $c \times c$ diagonal contingency table). Then the measure we compute is

$$D_{avg} = \frac{1}{|S| - 1} \left[\frac{\mathcal{F}_{\{w_{t_1}^{t_a}, w_{t_{a+1}}^{t_b}\}}}{\mathcal{F}_{max}} + \frac{\mathcal{F}_{\{w_{t_{b+1}}^{t_c}, w_{t_{c+1}}^{t_d}\}}}{\mathcal{F}_{max}} + \dots + \frac{\mathcal{F}_{\{w_{t_j}^{t_k}, w_{t_{k+1}}^{t_l}\}}}{\mathcal{F}_{max}} \right], \quad (7)$$

where $\mathcal{F}_{\{w_{t_a}^{t_b}, w_{t_{b+1}}^{t_c}\}}$ is the optimal objective function value obtained by clustering the pair of adjacent windows $w_{t_a}^{t_b}$, $w_{t_{b+1}}^{t_c}$. Observe that this criterion is different from the actual criterion optimized during the segmentation. D_{avg} compares our segmentation (which identifies dissimilar sets of clusters) to the case when there are exactly similar clusters. Lower values of this ratio indicate that the segmentation captures maximal independence between adjacent segments while higher values indicate the clusters obtained are more similar in adjacent segments.

3 Datasets and pre-processing

Our datasets came from a variety of sources. For each dataset described below, we retained only genes that have an annotation in the GO (Gene Ontology) biological process taxonomy (revision 4.205 of GO released on 14 March 2007), log transformed (base 10) their expression values and normalized them such that the mean expression of each gene across all time points is zero. The YMC1 dataset [10] consists of 36 time points collected over 3 continuous cycles. The original dataset consists of 6555 unique genes from the *S. cerevisiae* genome from which after preprocessing as described above we retained 3602 genes. We also analyzed another yeast metabolic cycle (YMC2) dataset [3] with 32 time points collected over 3 continuous cycles. (While in YMC1 [10] the authors claim that the cycle length is approximately 5 hours, in YMC2 [3], the authors claim that this cycle length is approximately 40 minutes.) Here we again retained 3602 genes. As our third dataset, we analyzed the well known yeast cell cycle dataset from experiments performed by Spellman et al. [8]. There are three components to the Spellman yeast cell cycle (YCC) data, following three different cell synchronization treatments with α -factor, cdc 15 and elutriation. We describe our analysis of the α -factor dataset that has 6076 genes with 18 time points over approximately 2 cycles. Our preprocessing results in a universal set of 2196 genes. Finally, we analyzed datasets from the experiments conducted by Shapira et al. [7], who studied the effects of oxidative stress induced by hydrogen peroxide (HP) and Menadione (MD) on the yeast cell cycle. We analyzed the datasets where HP and MD were added to the cells at 25 minutes after release from G1 arrest. The cells treated with HP were arrested in the subsequent G2/M phase while those treated with MD go through one cell cycle and were arrested in the G1 phase of next cycle. The HP dataset has 20 time points while the MD dataset has 14 time points. After preprocessing, we obtained a final set of 2471 genes in HP, and 2247 genes in MD datasets.

The above datasets are segmented using our dynamic programming algorithm. We used different settings for the numbers of clusters and different thresholds for minimum and maximum possible window lengths to search in the space of possible segmentations. Besides the number of clusters in each segment, and minimum/maximum constraints on window lengths, we parameterized the segmentation algorithm with a parameter λ that controls the sizes of the clusters in the resulting segmentations and can be adjusted to yield approximately equal cluster sizes. For YMC1, we experimented with the number of clusters in each segment ranging from 3 to 15, a minimum window length of 4 and maximum window length of 7, and $\lambda = 1.4$. For YMC2, the number of clusters is varied between 3 and 15, minimum and maximum window lengths used are 3 and 6 respectively, and $\lambda = 1.35$. For the YCC, we ranged from 3 to 15 clusters in each window

with a minimum window length of 3 and maximum window length of 5, and $\lambda = 1.25$. For segmenting the HP and MD datasets, the number of clusters is varied between 3 and 15, minimum and maximum window lengths used are 3 and 7 respectively, and $\lambda = 1.55$. After the segmentation reveals windows and clusters of genes in each window, we perform functional enrichment over the selected sets of genes. A hypergeometric p -value is calculated for each GO biological process term, and an appropriate cutoff is chosen using false discovery rate (FDR) q -level of 0.01 [9].

4 Comparisons with Biclustering

It is instructive to compare our segmentation and temporal process building approach with classical biclustering algorithms. Biclustering algorithms, even those that obey the temporal order of time points, will find groups of genes and groups of time points but will not yield a representation of the entire time course. A segmentation, on the other hand, is a partition of the entire time course. Consider for instance applying a biclustering algorithm such as e-CCC [5] and our segmentation algorithm to the yeast cell cycle data (see Figure 3 of our paper: time points 1–9). The classical metaphor for biologists to comprehend cell cycle data is to view it in terms of phases – something our algorithm naturally reveals. e-CCC, on the other hand, merely brings out a laundry list of biclusters which have to be processed to understand phases and transitions. But this is not possible and there are serious shortcomings:

Lack of a complete partition of the time course: e-CCC when run on the yeast cell cycle data would return results such as:

DNA replication initiation genes	3–5
strand elongation genes	3–7
metaphase activated genes	6–8

Note that because such an algorithm only looks for biclusters, the time points can overlap. Furthermore, the genes corresponding to these biclusters could overlap as well. Finally, and more importantly, some time points might not be represented in the answer (such as time points 1, 2, and 9, in the above case). These algorithms hence make no attempt to present an integrated picture of the temporal patterns manifest in the data, such as to find transition points in a global interval. Our segmentation algorithm goes considerably further. It posits that there are key time point boundaries around which genes re-group and re-organize. In addition, within a segment, we do not assume that there is a single process/function that is manifest but rather a group (set) of processes. It is these groups that get re-defined around the time point boundaries. For the above data, we get our result as:

cytokinesis	1–3
DNA replication	1–3
strand elongation	4–6
G1/S-specific transcription	4–6
mitotic spindle elongation	7–9
metaphase/anaphase transition	7–9

Note that all time points are represented in the answer but each time point participates in only one segment. Within each segment is enriched a set of processes. Using an entire segmentation such as this, we obtain an integrated view of the key phases underlying the dataset.

Inability to support temporal model building and inference: A more serious concern is that biclusters from algorithms such as e-CCC cannot be further processed into a complete temporal model, such as a finite state machine, Kripke structure, or other automata. This is a key contribution of our paper. Our segmentation and enrichment of each phase of the discovered segments immediately results in a Kripke structure (see Figure 1 of our paper). More importantly, we are able to combine such temporal models across experiments into an integrated process model (see Figure 2 of our paper). In summary, to use an oft-quoted metaphor, the biclustering algorithm gives us pieces of the puzzle which might not fit together and where some pieces are missing. The segmentation algorithm gives us pieces that fit together and which when composed yield the entire picture.

5 Gantt charts

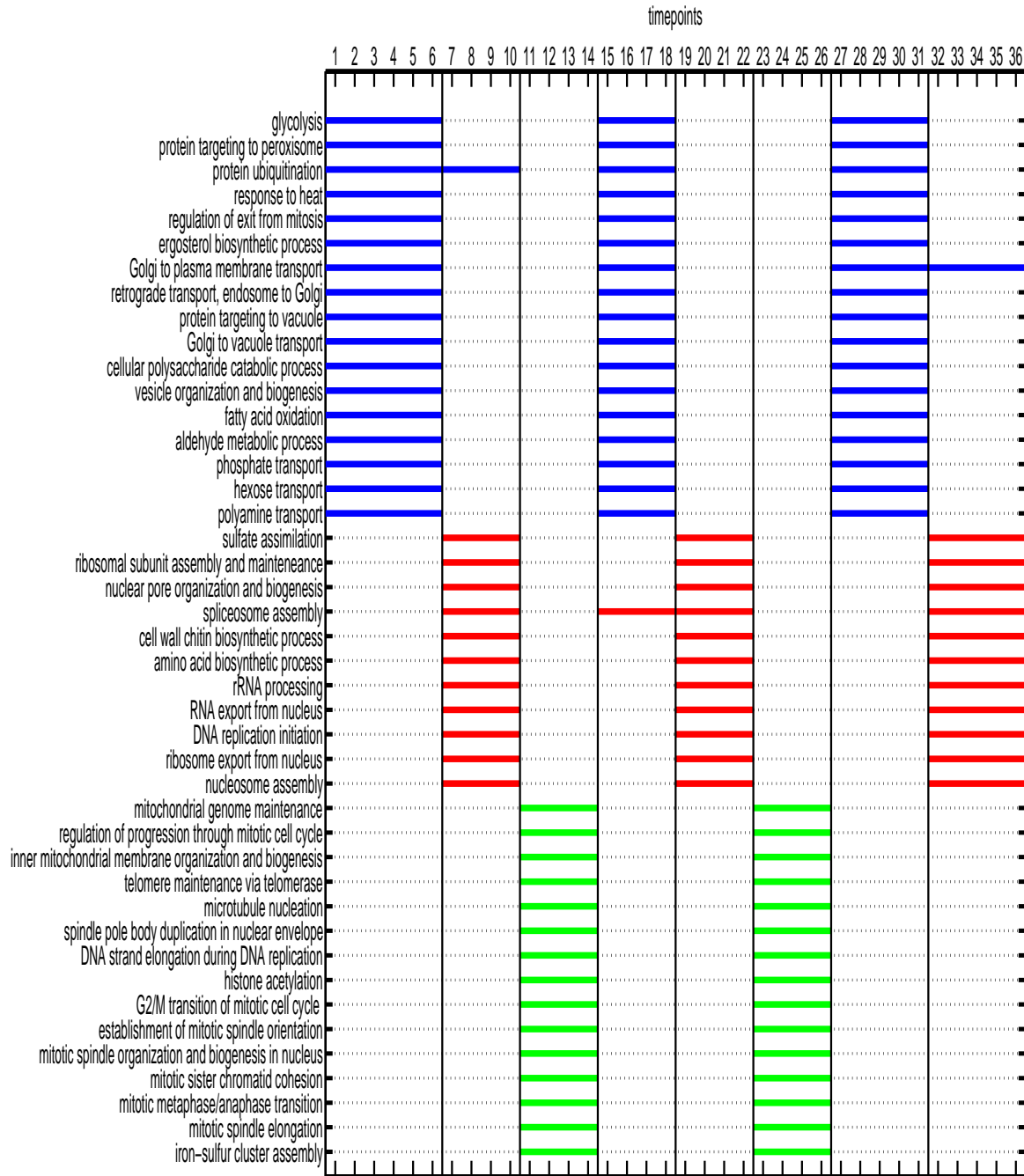


Figure 2: Gantt chart resulting from segmentation of the YMC1 dataset.

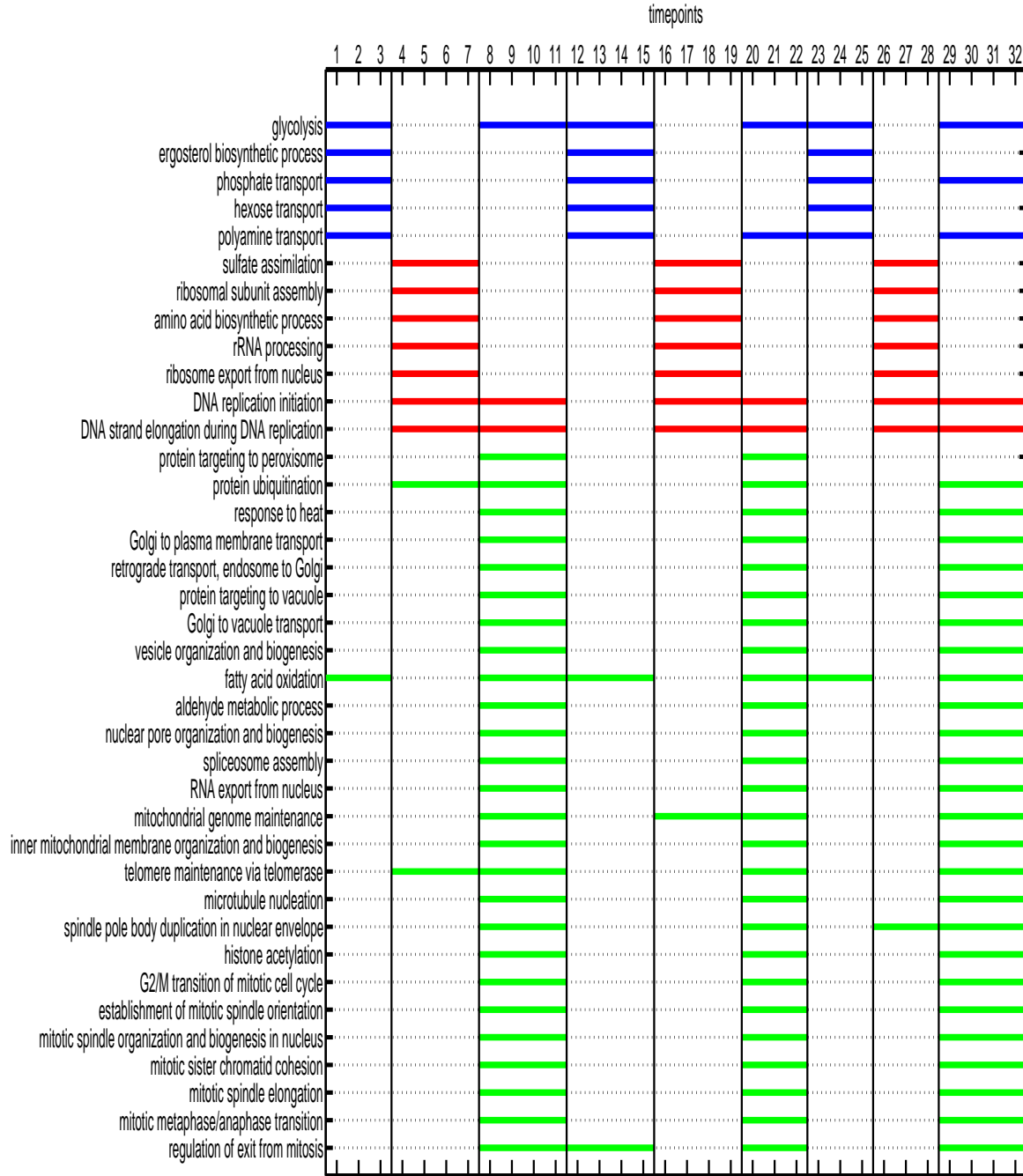


Figure 3: Gantt chart resulting from segmentation of the YMC2 dataset. Compare with Fig. 2.

References

- [1] J. F. M. Barthelemy and M. F. Riley. Improved multi-level optimization approach for the design of complex engineering systems. *AIAA J.*, 26:353–360, 1988.

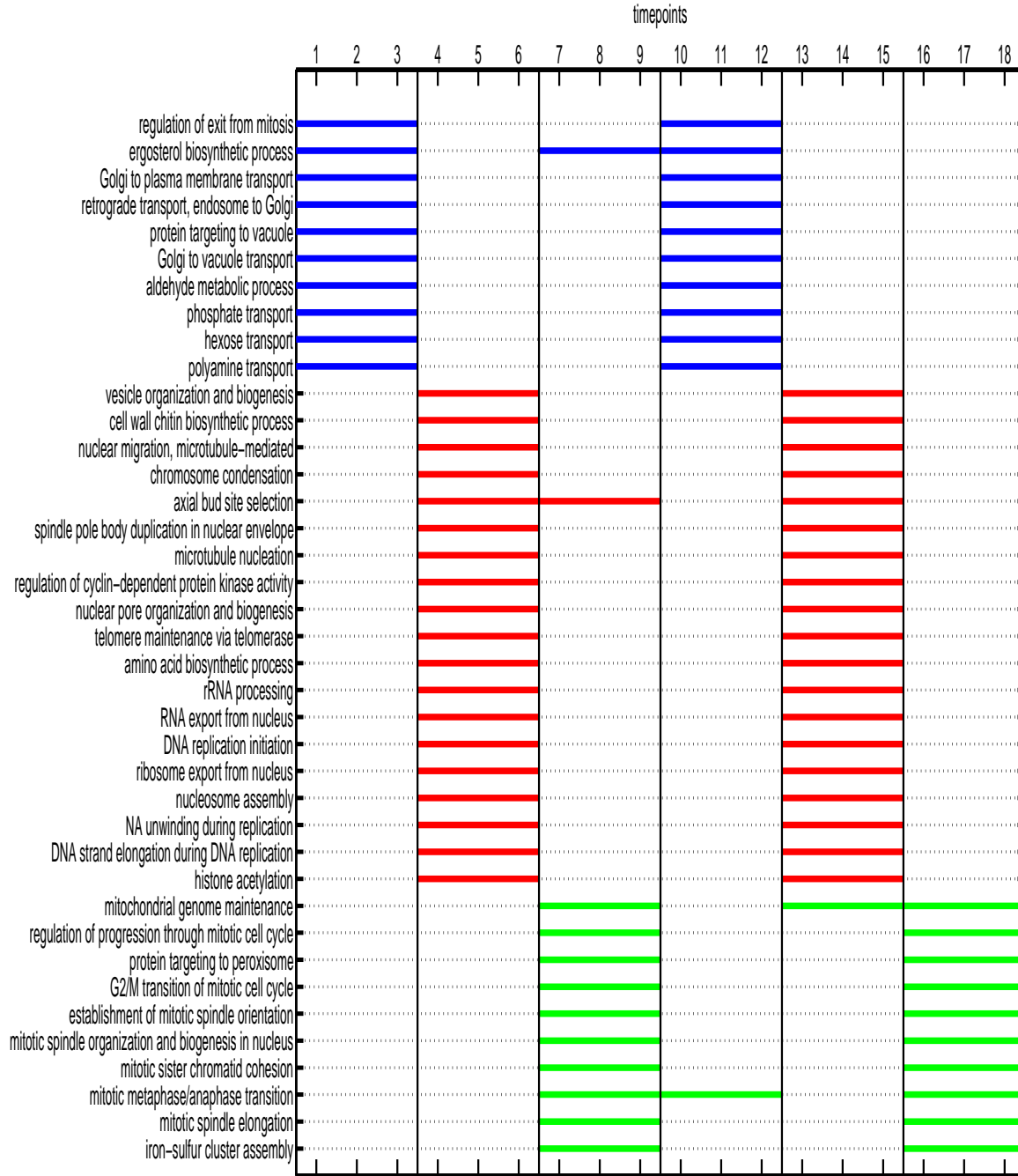


Figure 4: Gantt chart resulting from segmentation of the YCC dataset.

- [2] A.R. Conn, N.I.M. Gould, and P.L. Toint. *LANCELOT: A Fortran Package for Large-scale Nonlinear Optimization (Release A)*, volume 17. Springer Verlag, 1992.
- [3] R.R. Klevecz, J. Bolen, G. Forrest, and D. B. Murray. A genomewide oscillation in transcription gates dna replication and cell cycle. *PNAS*, 101(5):1200–1205, 2004.

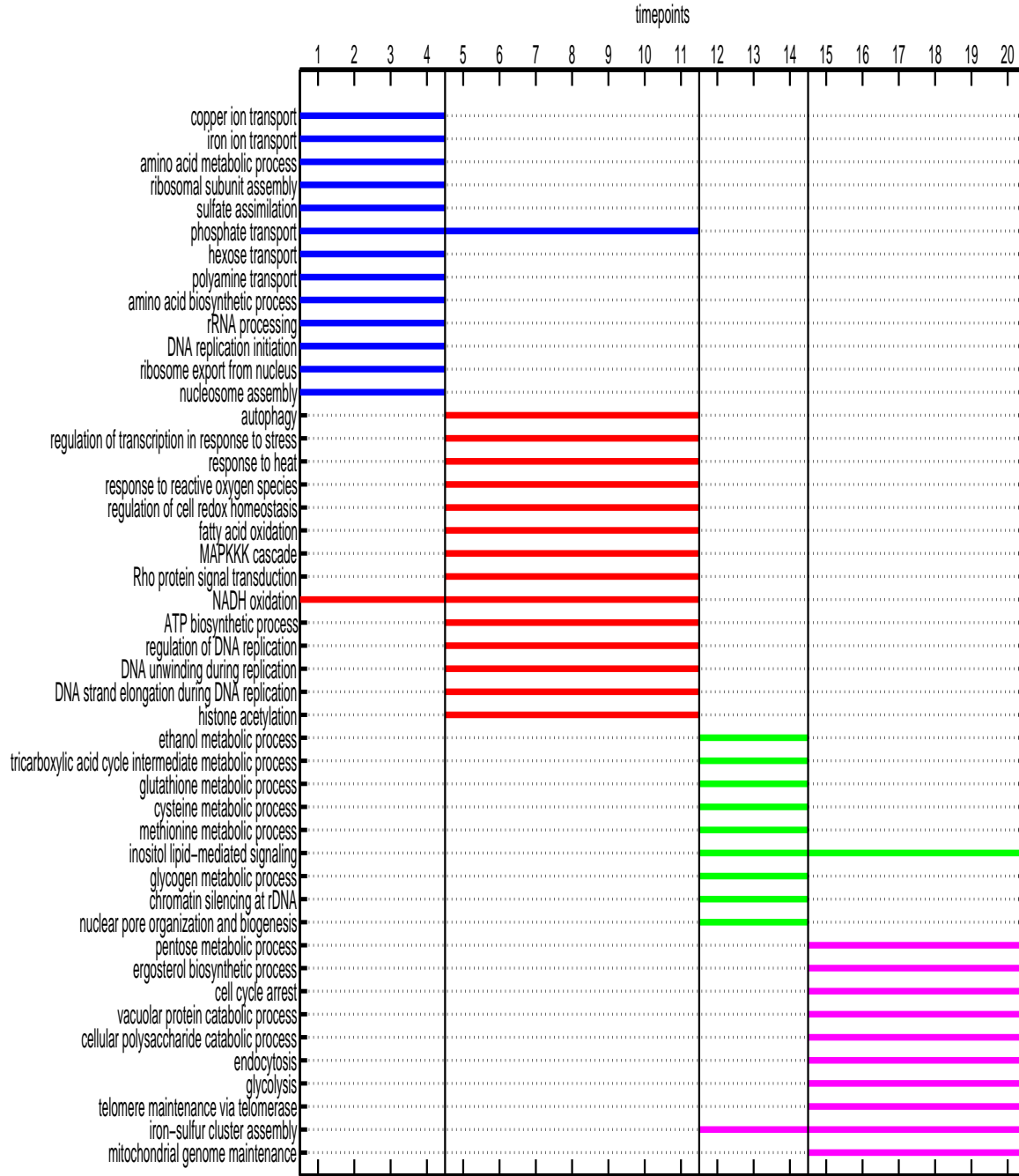


Figure 5: Gantt chart resulting from segmentation of the HP dataset.

- [4] G. Kreisselmeier and R. Steinhauser. Systematic control design by optimizing a vector performance index. In *IFAC Symp. on computer aided design of control systems*, pages 113–117, 1979.
- [5] S.C. Madeira and A.L. Oliveira. A Polynomial Time Biclustering Algorithm for Finding Approximate Expression Patterns in Gene Expression Time Series. *Algorithms for Molecular Biology*, Vol. 4, 2009.

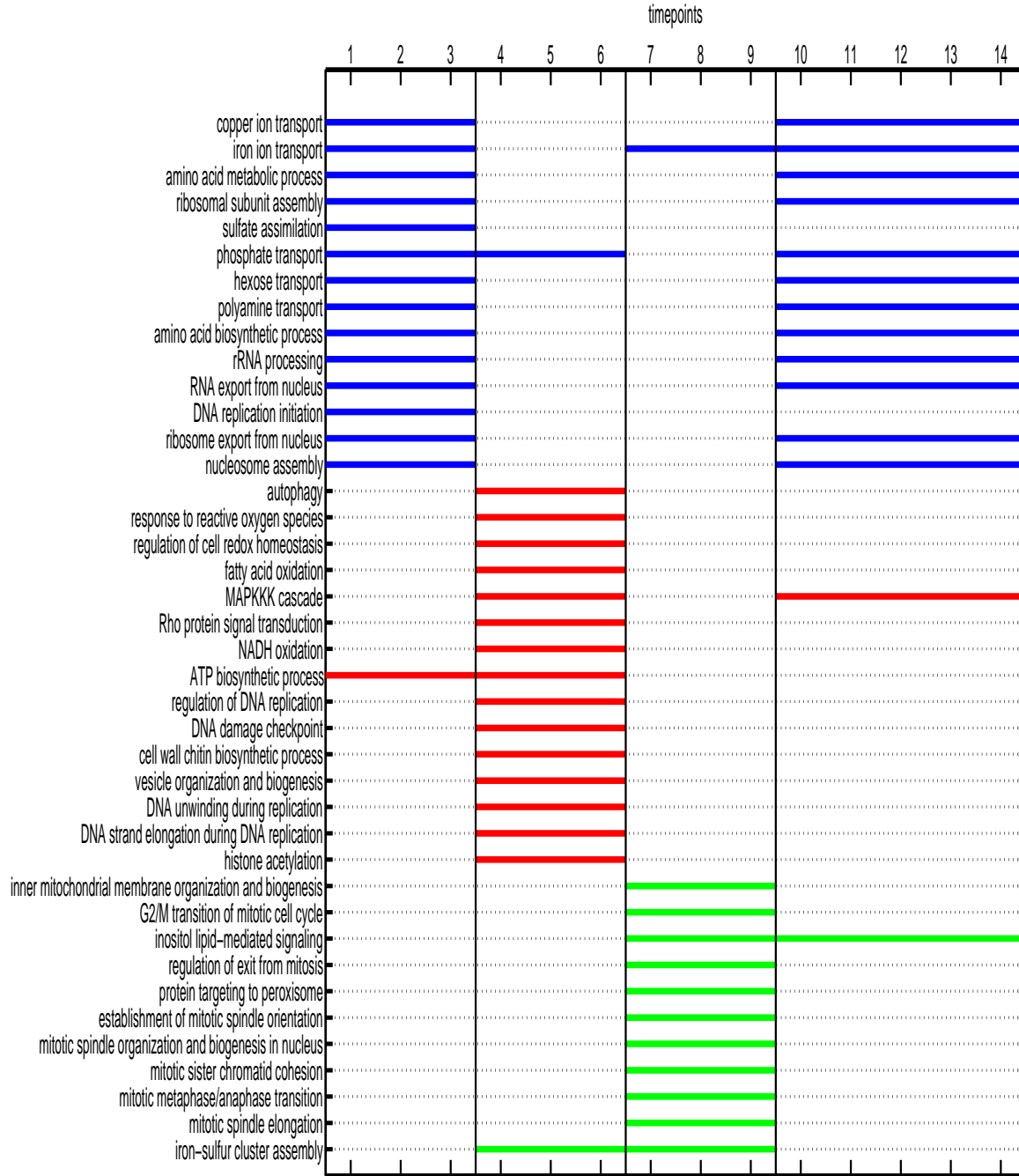


Figure 6: Gantt chart resulting from segmentation of the MD dataset.

- [6] R.L. De Mántaras. A Distance-Based Attribute Selection Measure for Decision Tree Induction. *Machine Learning*, 6(1):81–92, 1991.
- [7] M. Shapira, E. Segal, and D. Botstein. Disruption of Yeast Forkhead-associated Cell Cycle Transcription by Oxidative Stress. *Molecular Biology of the Cell*, 15(12):5659–5669, 2004.

- [8] P.T. Spellman, G. Sherlock, M.Q. Zhang, V.R. Iyer, K. Anders, M.B. Eisen, P.O. Brown, D. Botstein, and B. Futcher. Comprehensive Identification of Cell Cycle-Regulated Genes of the Yeast *Saccharomyces cerevisiae* by Microarray Hybridization. *Molecular Biology of the Cell*, 9(12):3273–3297, 1998.
- [9] J.D. Storey and R. Tibshirani. Statistical significance for genomewide studies. *PNAS*, 100(16):9440–9445, 2003.
- [10] B.P. Tu, A. Kudlicki, M. Rowicka, and S.L. McKnight. Logic of the Yeast Metabolic Cycle: Temporal Compartmentalization of Cellular Processes. *Science*, 310(5751):1152–1158, 2005.
- [11] L. T. Watson and R. T. Haftka. Modern homotopy methods in optimization. *Computer methods in appl. mechanics and engg.*, 74:289–305, 1989.