Abstract. In (Weinstein et al., 2015) the authors proposed a regulatory network model for both the cell cycle and cell differentiation of the vulva precursor cells (VPCs) of Caenorhabditis Elegans (CE) when the update schedule is parallel (synchronous). We extend this work by presenting efficient theoretical methods to extract cycle structures from the CE network in the sequential update case. We show that, by using a sequential update, the network can exhibit 125 distinct cycle structures. We comment on the variety and distribution of these cycle structure results. Furthermore, we present a novel approach to visualize node stability across all sequential updates. We introduce this approach with the motivation to aid modelers by providing further insight into the robustness of a biological network. Lastly, we formulate the theoretical tools required to extend these methods to the more general case of block sequential updates. The techniques presented in this work are general and can be used to analyze other network models.

1. Introduction

The nematode Caenorhabditis elegans (CE) has been extensively used as a model organism in research areas such as genetics, neuroscience, developmental biology, and cell differentiation [1]. In particular, the vulva of C. Elegans has been used as an experimental model for the study of organ formation, cellular fusion, and intracellular signaling [1].

In [1] the authors proposed a regulatory network model for both the cell cycle and cell differentiation of the vulva precursor cells (VPCs) of Caenorhabditis Elegans (CE) when the update schedule is parallel (synchronous). This network is presented in more detail in Section 3. The authors demonstrate that this proposed network model correctly exhibits results that are consistent with published experimental results and the authors allude to the inherent robustness of their network model.

The concept of network robustness is presented in [2] as a uniquitously observed and necessary property of biological systems. A dynamical property is said to be robust when it is not affected by small perturbations. In a real system, these perturbations model the noise which is intrinsic to reality, whereas in a model, perturbations refer to changes in the state variables (stability with respect to initial conditions) or to changes in the specification of the model itself [3]. In this study we will first consider modifications of the update schedule of the CE network. Update schedule modifications are changes to the model itself and therefore can cause variations in the dynamics of the system. In this study we only consider variations across different sequential update schedules. For more information on update schedules see [3].

Several biological network models have already been studied in the context of update schedule robustness such as the Lac Operon model [4], the Mammalian
cell cycle network model [5], and the Yeast cell cycle network model [6]. In these papers, the authors study update schedule robustness by analyzing dynamics properties such as the number of distinct cycle structures, periodic cycle lengths, and attraction basin sizes over all deterministic updating schemes.

We extend this work by presenting a computationally efficient method to extract these same cycle structure results. Specifically, we utilize the notion of a $\kappa$-class presented in the work of Mortveit and Macauley [7] to identify all cycle structure results in the CE network for the sequential update case. In Section 7 we extend this theoretical work to the more general block sequential update case.

Computationally efficient methods are important when studying large (or even small) networks, as these networks quickly become computationally intractable [8]. Nevertheless, larger and more complex important biological networks are emerging and efficient methods that allow us to study the update schedule robustness of these networks may be computationally feasible.

For example, in [9] the authors use a cell network for the interactions between the segment polarity genes in Drosophila Melanogaster. This single cell network consists of 14 nodes, and the authors consider a parasegment network model combining four of these cells, resulting in a network with $14 \times 4 = 56$ nodes. The update schedule robustness analysis of such a network is a costly task. As we will see in the context of the CE network, current robustness measures such as the number of distinct cycle structures and periodic cycle lengths may not be as insightful and illuminating as they were for characterizing the update schedule robustness of less complex networks. Furthermore, this analysis only provides feedback on the global dynamics of a biological network. In the context of a modeler determining a biological network containing many nodes and interactions, overall information on the robustness of the system may not be as important feedback as concentrated analysis on individual nodes.

In light of both these concerns, we introduce the idea of node update schedule stability. We define what this means formally in Section 5 using the activity notion from [10] and present node stability data in the context of the CE network under different sequential update schedules.

The paper is structured as follows: we present basic terminology and theory in Section 2 followed by a short overview of the Caenorhabditis Elegans (CE) network in Section 3. Computational results are presented in Sections 4 and 5 and extensions to block sequential updates in Section 7.

2. Background, Terminology, and Notation

We represent a network as a discrete dynamical system with a map of the form

$$F = (f_1, \ldots, f_n): K^n \rightarrow K^n,$$

where $K$ is some suitable finite set like $\{0,1\}$, see for example [11]. Associated to $F$ we have the dependency graph $G$ of $F$ whose vertex set is $V(G) = \{1, 2, \ldots, n\}$ and with edges all $(i,j)$ for which the function $f_i$ depends non-trivially on the variable $x_j$. We note that throughout this paper the terms vertex and node are used interchangeably. We will adopt the term node from [1] when referring to biological network models and adopt the term vertex in theoretical contexts.

With this representation, questions about an underlying biological system can be turned into precise questions about the dynamical system defined by Equation 1.
The states can be updated in parallel, as in the synchronous case, or through other asynchronous deterministic update schedules such as sequential or block sequential [6]. In this study we will only consider computational results for sequential updates. See [8] for more general information on sequential dynamical systems (SDSs). For SDS maps we denote the update permutation order by \( \pi \in S_G \) and represent \( F \) under this order as \( F_\pi \). In the following we give a brief overview of results found in [7].

In [7] the equivalence relation \( \sim_\alpha \) on \( S_G \) is introduced. Its equivalence classes are referred to as \( \alpha \)-classes. For any \( \pi \in S_G \) we can construct the \( \alpha \)-class, \([\pi]_G\), from the equivalence relation \( \sim_\alpha \). Where \( \pi \sim_\alpha \pi' \) implies the equality \( F_\pi = F_\pi' \). Furthermore, we can identify \([\pi]_G \in S_G/\sim_\alpha \) with a unique acyclic orientation \( O(\pi) \in \text{Acyc}(G) \). For more information on acyclic orientations see [7]. We define \( \alpha(G) \equiv |\text{Acyc}(G)| \) and the measure \( \alpha(G) \) is an upper bound for the number of functionally non-equivalent permutation SDS maps obtainable by varying the update order.

Two finite dynamical systems \( \phi: K^n \to K^n \) and \( \psi: K^m \to K^m \) are cycle equivalent if there exists a bijection \( h: \text{Per}(\phi) \to \text{Per}(\psi) \) such that
\[
\psi|_{\text{Per}(\phi)} \circ h = h \circ \phi|_{\text{Per}(\phi)},
\]

where \( \psi|_{\text{Per}(\phi)} \) and \( \psi|_{\text{Per}(\psi)} \) denote the restrictions of the maps to their respective sets of periodic points \( \text{Per}(\psi) \) and \( \text{Per}(\phi) \).

In [7] the equivalence relation \( \sim_\kappa \) is introduced. Its equivalence classes are referred to as \( \kappa \)-classes. For any \( O(\pi) \in \text{Acyc}(G) \) we can construct the \( \kappa \)-class, \([O(\pi)]_\kappa\), from the equivalence relation \( \sim_\kappa \). Specifically, for two orientations \( O, O' \in \text{Acyc}(G) \), \( O \sim_\kappa O' \) if \( O \) can be transformed into \( O' \) by a sequence of source-to-sink operations (clicks). It is shown in [7] that permutations that belong to \( \kappa \)-equivalent acyclic orientations induce cycle equivalent SDSs. We define \( \kappa(G) \equiv |\text{Acyc}(G)/\sim_\kappa| \), and the measure \( \kappa(G) \) is an upper bound for the number of distinct cycle equivalence classes obtainable through update order variations. The value \( \kappa(G) \) is presented as a complexity measure, and an enumeration method can be found in [7]. For computational efficiency, all \( \kappa \)-classes can be found by constructing a cycle-basis for use in the Coleman’s \( \nu \)-function (a complete invariant) which separates the acyclic orientations into their respective \( \kappa \)-classes [12].

Here, \( \alpha \)- and \( \kappa \)-equivalence are strictly combinatorial conditions based on the graph structure of a network model. However, they govern or restrict dynamics of corresponding asynchronous GDS maps and contribute a high amount of efficiency in dynamical analysis by vastly reducing the number of brute force computations. We now shift to computational analysis of the dynamics of a network model by introducing the cycle equivalence class, \([O(\pi)]_F\).

The cycle equivalence class is derived from computational results. It is clear from Equation 2 that two permutations are cycle equivalent if and only if they obtain the same cycle structure results from phase space computations. We refer to cycle structure results as the collection of all periodic \( n \)-cycles produced from a given phase space computation. In this paper we represent cycle structure results using a multiset. For example, a phase space computation that results in three 2-cycles and five 3-cycles is represented by the cycle structure multiset \([2(3), 3(5)]\).

**Definition 2.1.** The class \([O(\pi)]_F\) is constructed by taking the union of all distinct \( \kappa \)-classes that result in the same cycle structure as \( F_\pi \).
We let $\kappa_F(G)$ denote the total number of cycle equivalence classes, that is the total number of distinct cycle structures obtained for maps of the form $F_\pi$ with $\pi \in S_G$. It is also clear from construction that $\kappa_F(G) \leq \kappa(G)$.

### 3. The *Caenorhabditis Elegans* Network Model

In [1] the authors propose a regulatory network model for both the cell cycle and cell differentiation of the vulva precursor cells (VPCs) of *Caenorhabditis Elegans* (CE). This model is relatively new and is the first model to include the molecular mechanism involved in the control of the postembryonic cell cycle of CE. This new model is ideal for our study as we can showcase efficient methods to enumerate cycle structures (Section 4) and present new node stability data in Section 5 that may aid in the possible development of the CE network model.

In this section we give a brief overview of this model by citing excerpts from [1]. The regulatory network model consists of 14 nodes and 37 regulatory interactions (Figure 1). The set of vertex functions associated to this network can be found in [1].

![Figure 1. The network of molecules involved in the control of VPC fate determination and the cell cycle in CE. This figure is reproduced from [1].](image)

In our computational analysis we treat the states of the nodes LIN-3 and LS from Figure 1 as input parameters rather than node states. Also, we will not consider the LIN-39 node, as we will not simulate the loss of functions and therefore this node has no influence on the dynamics of the network. The state of the LIN-39 node can be directly derived from the MPK-1 node. Note that these changes do not affect the dynamics of the network model in any way.

In [1] the authors use a parallel update and interpret periodic cycles found as the patterns of molecular activation of the three vulval fates that cycle through the cell cycle. Specifically, the primary vulval fate is characterized by periodic cycles that exhibit a high level of LIN-39 and MPK-1 activation. The secondary fate is characterized by LIN-12i activation and the tertiary fate is characterized by a low
level of LIN-39 activation, and no LIN-12i, LIN-3, or lateral signal (LS) activation. For more information on vulval fates or the cell differentiation process see [1].

4. Preliminary results

We now report on cycle structure data from the CE network model and we let $G$ denote the dependency graph for this network model. In $G$ we have $\alpha(G) = 158,208$, $\kappa(G) = 5,312$, and $\kappa_F(G) = 125$. The efficiency of pre-computing $\kappa$-classes is clear here, as we can extract cycle structure results using only 5,312 update order representatives instead of 158,208. The ratio $\kappa_F(G)/\kappa(G) \approx 0.0235$ provides an initial measurement on the update order robustness of our model. A low ratio would suggest a high degree of cycle structure preservation across different update permutations, a general characteristic of network robustness.

In Figure 2 we showcase the variety of cycle structures in the CE network model by listing several cycle structure multisets found. The multisets found in Figure 2 correspond to the cycle structures of some 2197 $\kappa$-classes. The frequency of a cycle structure multiset is the number of $\kappa$-classes that resulted in the same cycle structure displayed by the multiset. We can then group the $\kappa$-classes that resulted in the same cycle structure to form a cycle equivalence class. Several statistics of multisets over all 5312 $\kappa$-classes are found in Figure 4.

<table>
<thead>
<tr>
<th>Cycle Structure Multiset</th>
<th>number of $\kappa$-classes in cycle equivalence class</th>
</tr>
</thead>
<tbody>
<tr>
<td>3(11) : 547</td>
<td></td>
</tr>
<tr>
<td>4(11) : 262</td>
<td></td>
</tr>
<tr>
<td>5(11) : 59</td>
<td></td>
</tr>
<tr>
<td>6(9) : 66</td>
<td></td>
</tr>
<tr>
<td>7(9) : 59</td>
<td></td>
</tr>
<tr>
<td>6(9) : 66</td>
<td></td>
</tr>
<tr>
<td>5(4) : 102</td>
<td></td>
</tr>
<tr>
<td>4(5) : 102</td>
<td></td>
</tr>
<tr>
<td>3(6) : 98</td>
<td></td>
</tr>
<tr>
<td>7(4) : 9(2) : 10(2)</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 2.** The results displayed represent the cycle structure multisets for 2197/5312 $\kappa$-classes.

To gain initial insight on the update robustness of the model, we can visualize how acyclic orientations are distributed across cycle equivalence classes. In Figure 3 we plot, in decreasing order, the percentage of total acyclic orientations contained in a corresponding cycle equivalence class. In a robust network model we expect this graph to be heavily front-loaded, as we want a large percentage of acyclic orientations to result in the same cycle structures.

From calculating the area under the curve in Figure 3, we find that around 75% of all acyclic orientations are distributed across 23 cycle structures. To further discuss results in the context of system robustness, let us consider the cycle structure results found in [1] when the update is a parallel one. The cycle structure multiset for the parallel update is $[9(4), 10(2), 11(2)]$. If we assume the model has robustness across update order variations, then we expect cycle structure multiset data to remain similar in the context of the biological network. Finding and interpreting these similarities can require a vast amount of insight on the biological system. We attempt to address a few similarities across all cycle structures found by referring to the statistics in Figure 4.
Figure 3. Distribution of total acyclic orientations across all cycle equivalence classes.

In Figure 4 we see that the total number of cycles within a multiset is roughly preserved from the parallel update scheme, as 61.1% of all \( \kappa \)-classes obtain cycle multisets of size 8 and 100% of all cycle multisets contain at least 8 total periodic cycles and at most 11 total periodic cycles. We can also see from the data that within a multiset there are at most 3 distinct \( n \)-cycle lengths where \( n \) is at least 3 and at most 10. Within a multiset consisting of at least 2 distinct \( n \)-cycle lengths (non-uniform multiset), the even multiplicity of \( n \)-cycles is fairly preserved from the parallel update, as all multiplicities are even in 82.0% of these cases. All these statistics contribute to several visual similarities across multiset data. In [5] the authors determine robustness from the “variety of limit cycles that a network can obtain depending on the updating scheme”. In the case of the CE network model, making a sufficient argument towards robustness using cycle structure data alone would be a difficult task because the “variety of limit cycles” is not exactly clear in the context of the biological system.

We claim this issue may arise in other biological network models, especially those with larger and complex graphs that may result in a large number of distinct cycle structures. This issue motivates the interest in collecting different and new forms of data from network models to provide further insight on update schedule robustness.
5. Results

Analyzing cycle structure is only a small part of characterizing the update robustness of a network model. We have shown that it is an efficient and economical way to provide insight into the robustness of a model, but there are several more important aspects of the network that need to be addressed. The main disadvantage of cycle structure analysis is that it alone cannot provide any information on the evolution of node states. This information is especially important, as the evolution of the node states characterize central aspects of the model such as cell differentiation and fate determination in the CE network model.

In [3] the authors state two characteristics of robustness, the stability with respect to initial conditions and network update schedule robustness. These two characteristics motivate their combined use as an interest of network model robustness. That is, how does node stability vary across different update schedules?

We suggest that analysis of this data may provide further insight into characterizing network model update schedule robustness.

To formally introduce and define the stability of nodes, we begin by defining activity. This function was first introduced by the group [5] in the parallel update case, but the definition can clearly be extended to any deterministic update schedule. In this paper we only consider sequential updates and we analyze short-term stability of dynamics through the activity function $\alpha_{F_\pi,i}: K^n \rightarrow \{0,1\}$ defined by [10]

$$\alpha_{F_\pi,i}(x) = I[ F_\pi(x + e_i) \neq F_\pi(x) ]$$

where $I$ is the indicator function and $e_i$ is the $i$th unit vector. In other words, $\alpha_{F_\pi,i}(x)$ measures if perturbing $x$ by $e_i$ results in a different successor state under $F_\pi$ than $F_\pi(x)$ [10].

**Definition 5.1.** The activity of $F_\pi$ with respect to node $i$ is the expectation value of $\alpha_{F_\pi,i}$ using the uniform measure on $K^n$:

$$\bar{\alpha}_{F_\pi,i} = E[\alpha_{F_\pi,i}] .$$

For a randomly chosen state $x \in K^n$, the value $\bar{\alpha}_{F_\pi,i}$ may be interpreted as the probability that perturbing $x_i$ will cause $F_\pi(x + e_i) \neq F_\pi(x)$ to hold. This activity notion may naturally be regarded as a measure of stability with respect to initial conditions [10]. Note that the activity function is ideally defined for nodes with boolean states, and we will not use it with respect to the ternary nodes in the CE network model.

**Definition 5.2.** Let $s \subseteq S_G$. We define $\bar{A}_{s,i} = \{ \bar{\alpha}_{F_\pi,i} \mid \pi \in s \}$ to be the activity of node $i$ over set $s$. If we plot $\bar{A}_{s,i}$ as a cumulative distribution function (CDF) then we define CDF$_{s,i}$ to be the activity CDF of node $i$ over set $s$.

For example, let us look at the LIN-12i node (second node, $i = 2$) in the CE network model. Let $[\pi]_G \in S_G/\sim_\alpha$ represent an $\alpha$-class in our network. It is clear that $A_{[\pi]_G,2}$ will result in a set with a single element (we prove this in a remark below for any alpha class). The activity CDF over $[\pi]_G$ is shown in blue in Figure 5.

Similarly, we can see activity distributions within the $\kappa$-class $[O(\pi)]_\kappa$ or the cycle equivalence class $[O(\pi)]_F$ by extracting a set of linear extensions from the orientations in the class.

We present the following universal characteristics of this data.
Figure 5. Within the same graph we have plotted the activity over a alpha class (blue), a kappa class (green), and a cycle equivalence class (red). Each class contains the permutation $\pi$.

Remark 5.3.  

(1) Activity over a $\alpha$-class is constant.  
(2) Activity over a $\kappa$-class is never guaranteed to be constant.

Proof. Permutations within a $\alpha$-class are functionally equivalent and therefore must have the same activity. Within a $\kappa$-class, transient states may evolve differently across different orientation permutations and therefore result in different activity measures. \qed

From section 2, we found 125 distinct cycle structures in the CE network model. Our original focus was to compare the activity CDFs across all cycle equivalence classes.

In Figure 6 we have plotted each of the 125 activity CDFs over cycle equivalence classes for the LIN-12i node. The use of cumulative distribution functions (CDFs) becomes an important tool to present this data because the use of CDFs prevent data biases due to the fact that the number of orientation permutations in each cycle equivalence class is not constant and can largely vary (see Figure 2).
Within the graph in Figure 6b we can see that some areas of the graph are densely populated with many CDFs intersecting in that region whereas some areas may be not be as heavily populated. This qualitative observation motivates the use of density plots. We form our density plots with the method outlined in Figure 7. The use of a density plot is strictly for visualization purposes. We extract density plots using a kernel density estimation package from Scipy. We note that in using this estimation, multi-modal distributions may be oversmoothed [13].

Figure 7. The construction of activity density plots.

In Figure 8 we create activity density plots for nodes in the CE network. Within these graphs, we also plot the activity of the node in the parallel (synchronous) update as a vertical pink line. We denote the parallel activity for a node $i$ by $\bar{a}_{P,i}$.

From Figure 8 we can visualize overlaps in activity CDFs over different sequential updates. In most of these plots the parallel update activity (pink line) coincides near higher areas of activity density. This is expected in robust networks due to the desire to preserve stability across different update orders. We note, in particular, the high activity correlations in the LIN-12i node data. In this node, we see the parallel update activity coincides almost exactly with the most dense point in Figure 8b. Within this density plot we also see clear overlaps in activity CDF data. This node is especially important in the context of the CE network because its activation and non-activation provide characterization for the representation of the secondary and tertiary fates of the vulval precursor cells. Therefore, we expect this node to exhibit a high amount of stability and robustness across different update orders. We also note that the predecessor nodes of LIN-12i (LIN-12m, CDK-2/CYE-1, and CDK-1/CYB-3) also share similar high activity correlations, which is expected.

Similarly, the MPK-1 node is another important node in the CE network as its presence characterizes the primary and tertiary fates of the vulval precursor cells. The activities of this node could not be computed due to its ternary state, but we can still vaguely infer stability from predecessor nodes. The only predecessor node of MPK-1 is LIN-12i, which we have already shown to have a high amount of stability.
(a) $\bar{a}_{FP, 1} \approx 0.141$

(b) $\bar{a}_{FP, 2} \approx 0.661$

(c) $\bar{a}_{FP, 3} \approx 0.438$

(d) $\bar{a}_{FP, 4} \approx 0.266$

(e) $\bar{a}_{FP, 5} = 1.0$

(f) $\bar{a}_{FP, 6} = 0.75$

(g) $\bar{a}_{FP, 7} = 0.625$

(h) $\bar{a}_{FP, 8} = 0.8125$
Further analysis of this type of data is a task for the modeler. We claim that stability analysis using methods presented may provide more insightful and illuminating feedback on the update robustness of a model than cycle structure analysis alone. For example, individual node stability analysis may be especially advantageous in network models that contain select nodes and interactions that have not been studied extensively in literature.

This presents some efficiency in the method as well, as stability computations only need to be carried out for select nodes. Furthermore, this method of stability analysis has no bias on the sizes of cycle equivalence classes. Thus we can extract similar, nearly identical, stability analysis data and observations by omitting CDF data computations from several large cycle equivalence classes.

6. Concluding Remarks

We have postulated that stability analysis of individual nodes over different update sequences may provide further insight into the robustness of a network model. Specifically, we address this analysis in the context of comparing activity distributions over all cycle equivalence classes. Unfortunately, this paper lacks rigorous arguments towards the validity and scope of this measure, i.e. we have not addressed, in detail, whether this measure provides significant contrast between robust and non-robust network models. We suggest that in comparing data over all cycle equivalence classes (rather than a subset) we create a larger, non-biased, contrast between robust and non-robust network models.

The general argument towards new forms of network robustness characterization data is clear, as cycle structure results may not be insightful arguments of network robustness and may provide little feedback and aid to biological model developers. In the use of block sequential updates this argument becomes even more clear, as we will see in the next section, the use of block sequential updates provides much more complexity to a network model and its cycle structure results.

The methods of data extraction used throughout this paper apply directly to the case of sequential update schedules. In the next section we formulate the theoretical tools required to extend these methods to the more general case of block sequential updates.
7. A Transversal for Block Update Sequences

In this section, we consider block sequential updates. More information on these can be found in [14]. Partitions of \( V(G) \) are denoted by \( \mathcal{B} = (B_1, \ldots, B_k) \), and the elements of \( \mathcal{B} \) are called blocks.

**Definition 7.1.** A block \( B \) is *indecomposable* if the subgraph in \( G \) induced by \( B \) is connected. A partition \( \mathcal{B} \) is indecomposable if all of its blocks are indecomposable.

Let \( G \) be the dependency graph of a GDS, or the dependency graph of the sequence of functions \((f_i)_{i=1}^n\). For the purpose of determining when \( F_{\pi(B)} \) equals \( F_{\pi'(B')} \) it is sufficient to consider the case where \( G \) is undirected [8].

**Lemma 7.2.** Every map of the form \( F_{\pi(B)} \) can be obtained as \( F_{\pi'(B')} \) where \( B' \) is indecomposable.

**Proof.** Any decomposable block \( B \) in \( \mathcal{B} \) is the union of indecomposable blocks \((B_i')_{i=1}^m\) of \( B' \). It follows that

\[
F_B = F_{B_1'} \circ \cdots \circ F_{B_m'}
\]

where the order does not matter by the definition of indecomposability. \( \square \)

**Definition 7.3.** Let \( \mathcal{B} \) be a partition of \( V(G) \). The *block graph* of \( G \) and \( \mathcal{B} \), denoted by \( G \setminus \mathcal{B} \), has vertex set \( \mathcal{B} \) and there is an edge between \( B_i \) and \( B_j \) whenever these blocks contains at least one pair of vertices that are connected in \( G \). One may take \( G \setminus \mathcal{B} \) to be directed if \( G \) is directed.

For a fixed partition \( \mathcal{B} \) of \( G \), we may extend the equivalence relation \( \sim_\alpha \) on \( S_G \) from [7] to \( G \setminus \mathcal{B} \) and \( S_G \).

**Lemma 7.4.** There is a bijection \( \phi : S_{V(G \setminus \mathcal{B})} / \sim_\alpha \rightarrow \text{Acyc}(G \setminus \mathcal{B}). \)

Here \( \sim_\alpha \) is defined as before.

**Proof.** From Definition 7.3, it is clear we can create combinatorial graphs using connected blocks in the same way we created graphs using connected vertices. Notice that any permutation \( \pi = (B_1, \ldots, B_n) \in S_G \) must still induce a linear ordering on the block set, so therefore we obtain a map

\[
\phi' : S_{V(G \setminus \mathcal{B})} \rightarrow \text{Acyc}(G \setminus \mathcal{B})
\]

It is clear that we may extend ideas from [7] and create an analogous update graph \( U(G \setminus \mathcal{B}) \). To show that we have a well-defined map, it is sufficient to consider the case with two adjacent block permutations \( \pi \) and \( \pi' \) in \( U(G \setminus \mathcal{B}) \). The general case will then follow by induction on the length of the path connecting \( \pi \) and \( \pi' \). If \( \pi \) and \( \pi' \) are adjacent, they differ in exactly two consecutive entries, and the corresponding entries are not connected by an edge in \( G \setminus \mathcal{B} \). Consequently, we must have \( \phi'((\pi)) = \phi'((\pi')) \), thus \( \phi' \) is constant on each equivalence class.

To show that \( \phi \) is a bijection, let \( \mathcal{O}_{G \setminus \mathcal{B}} \) be an acyclic orientation. We may extend ideas from [7] and construct an analogous block canonical permutation \( \hat{\pi} \) of some given equivalence class \( [\pi]_{G \setminus \mathcal{B}} \). All the arguments from [7] follow, and the proof of the bijection is complete. \( \square \)

**Lemma 7.5.** Let \( G \) be the dependency graph of \((f_i')_i\). If \( \mathcal{B} \) and \( \mathcal{B}' \) are distinct indecomposable partitions of \( V(G) \), then there are non-trivial functions \((f_i)_i\) with \( G \) as dependency graph such that \( F_{\pi(B)} \neq F_{\pi'(B')} \) for any permutations \( \pi \) and \( \pi' \) of the respective partitions \( \mathcal{B} \) and \( \mathcal{B}' \).
Proof. Since $\mathcal{B}$ and $\mathcal{B}'$ are distinct at least one of them, say $\mathcal{B}$, has a block $B$ of size $\geq 2$ not contained in the other partition $\mathcal{B}'$. As a result, there are connected vertices $v, v' \in B$ that belong to distinct blocks $B_v$ and $B_v'$ of $\mathcal{B}'$. We may assume that the block containing $v$ in $\mathcal{B}'$ occurs before the one containing $v'$ for the sequence $\pi'$. Let $x \in \{0, 1\}^n$ be the state for which $x_w = 1$ for all vertices $w$ proceeding $v$ in $\pi(\mathcal{B})$ and with all other vertex states being 0. We assign (nor$_i$)$_i$ functions to all vertices in $G$. Under $\pi(\mathcal{B})$ we have $(F_{\pi(\mathcal{B})}(x))_v = (F_{\pi(\mathcal{B})}(x))_{v'} = 1$. However, under $\pi'(\mathcal{B}')$ it is clearly impossible for both $v$ and $v'$ to have state 1 in $F_{\pi'(\mathcal{B}')}$. 

Lemma 7.6. Let $G$ be the dependency graph of $(f_i^\pi)_1$. If $\mathcal{B}$ is an indecomposable partition of $V(G)$, and permutations $\pi$ and $\pi'$ belong to different components in $U(G \setminus \mathcal{B})$, then there are non-trivial functions $(f_i^\pi)$ with $G$ as dependency graph such that $F_{\pi(\mathcal{B})} \neq F_{\pi'(\mathcal{B})}$.

Proof. For two different components in $U(G \setminus \mathcal{B})$, the update order between at least one connected block pair $(B_i, B_j)$ must have been switched. Let $B_i$ proceed $B_j$ under $\pi(\mathcal{B})$.

Because $B_i$ and $B_j$ are connected, by definition there must exist at least one pair of connected vertices $(v, v')$ where $v \in B_i$ and $v' \in B_j$. Assign (nor$_i$)$_i$ functions to vertices $v$ and $v'$. Assign (or$_i$)$_i$ functions to each other vertex in $G$. Let $x \in \{0, 1\}^n$ be the state for which all vertices $x = 0$. Under $\pi(\mathcal{B})$ each vertex $i$ in blocks proceeding or equal to block $B_i$ will be updated by $(F_{\pi(\mathcal{B})}(x))_i = \text{or}_i(0, \ldots, 0) = 0$. Vertex $v$ will be updated by $(F_{\pi(\mathcal{B})}(x))_v = \text{nor}_v(0, \ldots, 0) = 1$. Because $v'$ appears in a later block $B_j$ and depends on $v$, it must be updated by $(F_{\pi(\mathcal{B})}(x))_{v'} = \text{nor}_{v'}(1, \ldots, 0) = 0$.

Similarly, under $\pi'(\mathcal{B})$ each vertex $j$ in blocks proceeding or equal to block $B_j$ will be updated by $(F_{\pi'(\mathcal{B})}(x))_j = \text{or}_j(0, \ldots, 0) = 0$, so $v'$ will be updated by $(F_{\pi'(\mathcal{B})}(x))_{v'} = \text{nor}_{v'}(0, \ldots, 0) = 1$. Therefore $(F_{\pi(\mathcal{B})}(x))_{v'} \neq (F_{\pi'(\mathcal{B})}(x))_{v'}$.

Theorem 7.7. The set of all block update sequence equivalence classes over $G$ is in a bijective correspondence with

$$\bigcup_{\mathcal{B} \in \mathfrak{B}} \text{Acyc}(G \setminus \mathcal{B}) .$$

We can therefore obtain a transversal of block update sequences by taking the canonical permutations belonging to all the block graphs $G \setminus \mathcal{B}$.

Proposition 7.8. For any $\mathcal{B} \in \mathfrak{B}$ and permutation $\pi \in S_G$, the maps $F_{\sigma_k(\pi(\mathcal{B}))}$ and $F_{\pi(\mathcal{B})}$ are cycle equivalent.

Proof. We extend the notation from [7] to denote $\sigma_k(\pi(\mathcal{B}))$ by the block permutation obtained from $\pi(\mathcal{B})$ by cyclically shifting $\pi(\mathcal{B})$ to the left $k$ times. The complete proof is identical to the one presented in [7], in which we substitute block permutations for the regular permutation case.

From Proposition 7.8 it is clear that we may extend the equivalence relation $\sim_\kappa$ on $\text{Acyc}(G)$ from [7] to $\text{Acyc}(G \setminus \mathcal{B})$. For any $\mathcal{B} \in \mathfrak{B}$ we can construct all $\kappa$-classes, $\text{Acyc}(G \setminus \mathcal{B})/\sim_\kappa$, and furthermore, extract all cycle structures from the set $\bigcup_{\mathcal{B} \in \mathfrak{B}} \text{Acyc}(G \setminus \mathcal{B})/\sim_\kappa$. The size of this set presents an upper bound for the
number of cycle equivalence classes obtainable through block sequential update order variations. It remains to show that this upper bound is sharp.

**Conjecture 7.9.** Let $G$ be the dependency graph. Then there exist a set of functions $(f_i)$, with $G$ as the graph such that the set of all block update sequence cycle equivalence classes over $G$ is in a bijective correspondence with

$$\bigcup_{\mathcal{B} \in \mathcal{B}} \text{Acyc}(G \setminus \mathcal{B})/\sim_{\kappa}.$$ 

We can therefore analogously use all the same methods used in the sequential case in section 2 to extract cycle equivalence classes in the block sequential case. That is, we first generate all the distinct indecomposable partitions of $V(G)$ and use Theorem 7.7 to find all block update sequence equivalence classes. We then construct a set of all cycle equivalence classes from $\bigcup_{\mathcal{B} \in \mathcal{B}} \text{Acyc}(G \setminus \mathcal{B})/\sim_{\kappa}$ by computing the phase spaces of block sequence representatives within this set. Finally, we may analyze results using the same methods used in Sections 4 and 5.

**Acknowledgments**

We thank our external collaborators and members of the Network Dynamics and Simulation Science Laboratory (NDSSL) for their suggestions and comments. This work has been partially supported by DTRA Grant HDTRA1-11-1-0016.

**References**


